TRABAJO FIN DE CARRERA

TÍTULO DEL TFC: Medida y análisis de matrices de tráfico en redes IP

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Telemática

AUTOR: Iván Minguillón de la Colina

DIRECTOR: David Rincón Rivera

FECHA: 9 de enero de 2011

Título: Medida y análisis de matrices de tráfico en redes IP

Autor: Iván Minguillón de la Colina

Director: David Rincón Rivera

Fecha: 9 de enero de 2011

Resumen

Para diseñar, planificar y administrar una red de telecomunicaciones de manera eficiente, se requiere una matriz de tráfico. Ésta describe la cantidad de información que circula entre cualquier par de nodos de entrada y salida de una red en un cierto periodo de tiempo. Algunas de las aplicaciones de las matrices de tráfico son la detección de anomalías, balanceo de carga o encaminamiento, aplicando optimización de rutas para evitar congestiones.

Para un operador, medir de forma directa la matriz de tráfico implica retos técnicos y organizativos. Debido a esto, muchos investigadores han dedicado esfuerzos para encontrar un método con el que sea posible inferir una matriz a partir de las medidas de carga de los enlaces; a día de hoy uno de ellos ya ha demostrado resultados prometedores: el método de Tomogravedad. En el presente trabajo se aplica este método para generar las matrices de tráfico estimadas de las redes Abilene y GÉANT, así como calcular su error comparándolas con las matrices originales, ya que éstas últimas son de dominio público.

Por otra parte, RedIRIS, la red académica española, nos proporcionó información de NetFlow y datos SNMP de los enlaces con el objetivo de calcular la matriz de tráfico de su red. Para conseguir generar un modelo de matriz de tráfico, se necesita analizar previamente los datos aportados utilizando herramientas específicas de tratamiento de flujos de red y contadores SNMP, como son *nfdump, NfSen, flow-tools* y *RRDtool*. En este estudio se recogen los pasos de instalación a partir del código fuente de todas las aplicaciones mencionadas, el formato de los archivos de cada una de ellas, los comandos necesarios para procesar las trazas y los resultados obtenidos. Este trabajo supone el punto de partida para que futuros estudios consigan modelar y predecir la matriz de tráfico de RedIRIS.

Title: Measurement and analisys of traffic matrices on IP networks

Author: Iván Minguillón de la Colina

Director: David Rincón Rivera

Date: January, 9th 2011

Overview

In order to efficiently carry on the tasks related to the design, plan and management of a telecommunications network, a traffic matrix is needed. A traffic matrix describes the volume of traffic exchanged by any pair of ingress and egress nodes in a network during a certain period of time. Some applications of traffic matrices are anomaly detection, load balancing or route optimization in order to avoid network congestion.

Measuring the matrix directly entails setting up an additional infraestructure to get the raw data, which is both expensive, organizationally complex and unfeasible in most cases. That is the main reason why many researchers have spent efforts in order to find inference methods able to estimate a traffic matrix from link load measurements, which are much easy to obtain. One of them, the Tomogravity method, has shown promising results. In this thesis we apply this method to traffic matrices obtained from Abilene and GÉANT networks, and measure the error comparing them to the original matrices, which are already released as public domain data.

On the other hand, the Spanish education and research network RedIRIS provided NetFlow and SNMP data from their links in order to model their traffic matrix. To generate a traffic matrix, all data has to be analyzed using flow programs and SNMP counters information tools, such as *nfdump, NfSen, flow-tools* and *RRDtool.* All the steps regarding software installation from source code of every application are compiled in this thesis, with details about commands to deal with the data and the results obtained. This thesis is a starting point for future works in which to model RedIRIS traffic matrix.

ÍNDICE

INTRODUCCIÓN

Algunos de los problemas que la ingeniería de tráfico en redes IP aborda son la configuración de protocolos de encaminamiento, el dimensionado de red, estrategias de recuperación en caso de fallos, balanceo de carga u optimización de rutas en los enlaces troncales para evitar congestiones y fallos en el servicio. Una matriz de tráfico proporciona, para cada nodo de entrada y salida de la red, el volumen de tráfico transportado entre ellos durante un cierto periodo de tiempo. La ingeniería de tráfico utiliza las matrices para diagnosticar y gestionar la congestión de red, haciendo de ellas un elemento clave para el diseño y planificación de una red de telecomunicaciones.

Una forma de determinar la matriz de tráfico es a través de mediciones directas, recogiendo los flujos de datos en los puntos de entrada. Para ello se requiere una infraestructura adicional en los routers, lo que provoca un consumo extra de CPU en dichos dispositivos que puede derivar en una reducción en la capacidad de encaminamiento, ya que los enlaces troncales de las grandes redes de comunicaciones transportan terabytes de datos al día. Es en este punto donde se hizo evidente la necesidad de inferir la matriz de tráfico a partir de otra información disponible y no tan costosa de obtener, como son los contadores SNMP, la información de encaminamiento o la topología de red. En este trabajo se aplica a las redes Abilene y GÉANT uno de esos métodos, Tomogravedad, y se calcula el error producido por la inferencia de la matriz de tráfico.

RedIRIS, la red nacional española de investigación y docencia, nos proporcionó datos sobre el volumen de información transmitida en su red con el objetivo de calcular la matriz de tráfico. Toda esa serie de trazas se procesan mediante las herramientas *flow-tools*, *nfdump* y *RRDtool*, consiguiendo así las transformaciones de formato necesarias para llegar a generar la matriz de tráfico.

El resto del documento se organiza tal como se describe a continuación. En el primer capítulo se definen las matrices de tráfico, se explica su importancia, varios modos de obtención y se presenta uno de los métodos para inferirlas, el método Tomogravedad. En el segundo capítulo se aplica Tomogravedad en las redes Abilene y GÉANT. Se comprueba la eficacia de este método de inferencia mediante la comparación de las matrices originales con las estimadas. En el tercer capítulo se presentan las aplicaciones con las que se analizarán y procesarán los datos proporcionados por RedIRIS. En el cuarto capítulo se describen las mediciones realizadas en RedIRIS y se analizan y procesan los datos de red con el objetivo de llegar a modelar la matriz de tráfico, detallando el contenido de las trazas y su formato. Para finalizar, se presentan las conclusiones a las que se ha llegado junto con las líneas futuras de investigación.

CAPÍTULO 1. Matrices de tráfico

Parte de la información aquí expuesta sobre matrices de tráfico proviene de los TFC de Javier Torres [20] e Isaac Balasch [2].

1.1. Definición

Las matrices de tráfico (TMs, por su abreviatura en inglés, *Traffic Matrices*) describen el volumen de tráfico que circula entre los nodos de ingreso y egreso de una red, los cuales pueden ser nodos simples (routers) o Puntos de Presencia (PoPs), de donde cuelgan otros nodos o redes enteras. Para cada par de nodos (entrada-salida) la matriz de tráfico especifica el volumen de tráfico transportado durante un cierto periodo de tiempo. En los conjuntos de datos utilizados en este trabajo, este periodo de tiempo es de cinco minutos para Abilene[1] (la red académica norteamericana) y quince para GÉANT[2] (la red académica europea).

Las matrices de tráfico no tienen necesariamente que ser simétricas, ya que la distribución del tráfico en una red tampoco lo es. Un caso especial será la generación de tráfico de un nodo hacia él mismo; esto es debido a que en las matrices de tráfico existen relaciones por un origen X con un destino también X, esto es, un par X-X, por ejemplo Washington-Washington. Esto es aceptable en el caso de que un nodo sea un Punto de Presencia, ya que habrá tráfico generado por los propios clientes conectados directamente a ese PoP que tendrá como destino clientes del mismo PoP.

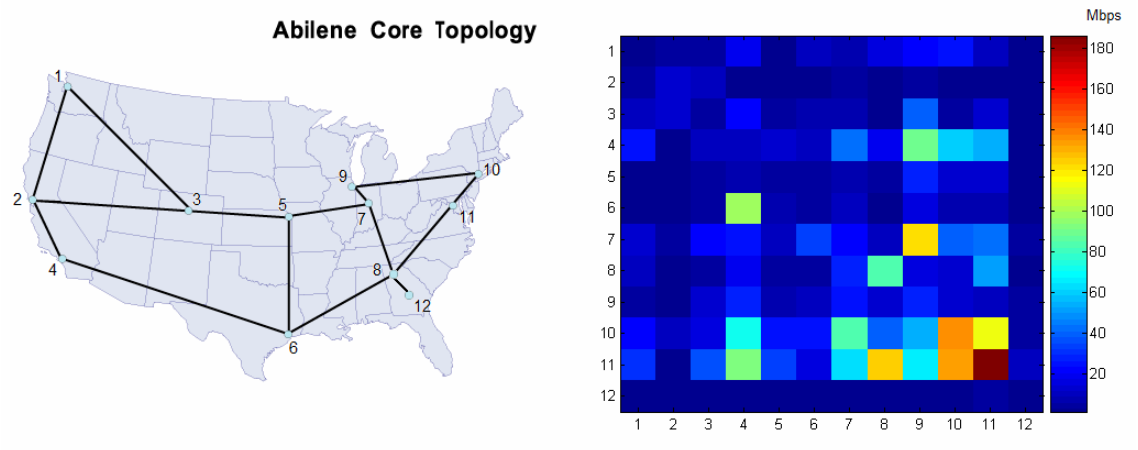


Fig. 1.1 Topología de Abilene y representación gráfica de una de sus matrices de tráfico (01/03/2004 de 00:00 a 00:05)

[1] Abilene Network, http://www.internet2.edu/network
[2] GÉANT Network, http://www.geant.net

En las figuras 1.1 y 1.2 se puede comprobar el resultado de representar una matriz de tráfico como si fuera una imagen, con origen en el eje vertical y destino en el horizontal.

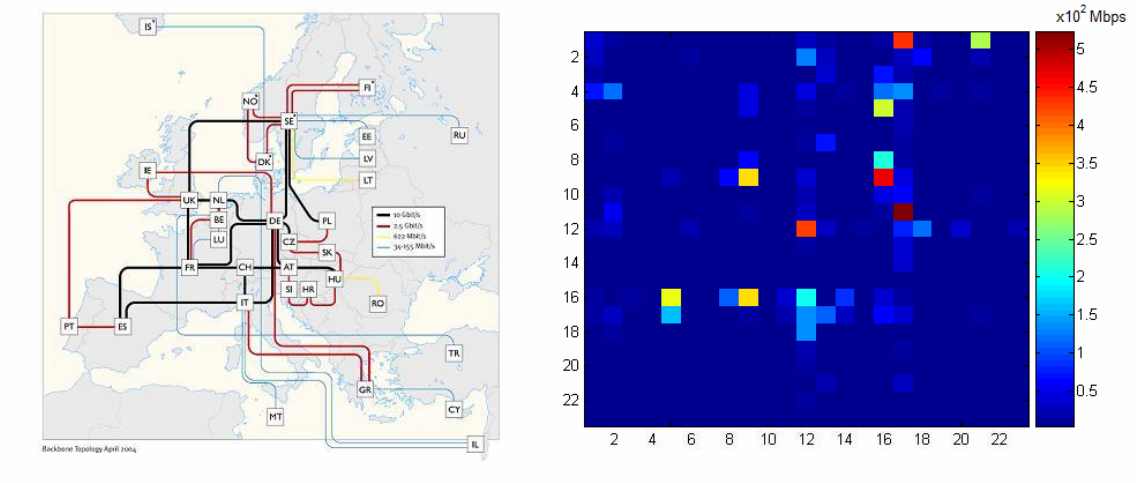


Fig. 1.2 Topología de GÉANT y representación gráfica de una de sus matrices de tráfico anonimizadas (01/04/2005 de 20:45 a 21:00)

En el capítulo 4 se analizan las trazas proporcionadas por RedIRIS [15], la red académica española, para generar su matriz de tráfico. RedIRIS es uno de los ejemplos de redes nacionales de investigación y docencia (NREN, *National Research and Education Networks*) que forman parte de GÉANT; como se puede ver en la figura 1.3, su topología es mallada y comunica las instituciones universitarias y de investigación españolas entre sí, a través de las redes académicas autonómicas.

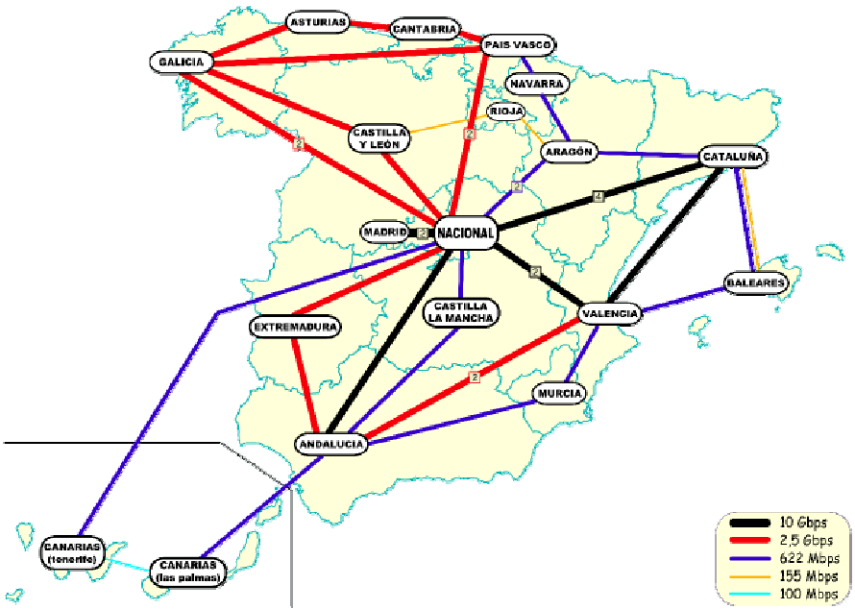


Fig. 1.3 Topología de RedIRIS

1.2. Importancia de las matrices de tráfico

Las matrices de tráfico relacionan tres parámetros básicos de una red: origen, destino y volumen de tráfico, resultando de gran utilidad en la gestión y planificación de redes IP.

La disponibilidad de buenos modelos es esencial para que los operadores de red puedan obtener unas estimaciones correctas de las matrices y sean capaces de aplicarlas en tareas relacionadas con la ingeniería de tráfico, incluyendo configuración de protocolos de encaminamiento, estrategias de recuperación de fallos, balanceadores de carga, aprovisionamiento o dimensionado. La información sobre el tamaño y la localización de flujos es decisiva para la planificación del crecimiento de las redes y el diagnóstico de problemas. Construir una red IP troncal apoyándose en el conocimiento ofrecido por las matrices de tráfico es crucial para poder diseñarla eficientemente.

Con la información que proporcionan los propios operadores es posible diseñar una nueva red, detectar anomalías, simular la cantidad de información que se transportará y predecir el comportamiento de los enlaces que conforman la red en un instante determinado. Éstas son algunas de las razones por las que hay un gran interés dentro de los grupos de investigación de la ingeniería de tráfico sobre el modelado de las matrices de tráfico y su predicción.

1.2.1. Aplicaciones de las matrices de tráfico

Los operadores de red siempre han buscado --y continúan haciéndolo--, obtener mejores modelos de estimación o soluciones más precisas a los problemas que el flujo de tráfico sobre un enlace puede producir: planificación, fallos, obtención de información para la monitorización, detección a priori de fluctuaciones impredecibles, etc. Estos son los principales objetivos en un análisis de matrices de tráfico:

- Síntesis: uno de los primeros problemas existentes es la gran cantidad de muestras de matrices de tráfico necesaria para determinar simulaciones fiables, por ejemplo, de una nueva red o protocolo. Una vez obtenida una síntesis de TM se pueden crear matrices de tráfico con las que diseñar y crear redes enteras (localización de routers o PoPs, enlaces y asignación de capacidades). La síntesis ha de ser tan fiable como sea posible, con el objetivo de minimizar el error entre ésta y las muestras originales [14].

- Routing y balanceo de carga: otra importante aplicación de las matrices de tráfico es la de optimizar el encaminamiento. Uno de los objetivos de la ingeniería de tráfico es la optimización mediante balanceo de carga, calidad de servicio (QoS), etc. En ciertas situaciones se puede dar la

existencia de enlaces con una gran carga agregada, provocando así una congestión en la red. Una forma eficiente de dar solución a este problema podría ser introducir rutas alternativas para balancear el tráfico hacia otros nodos de la red. Por este motivo es importante tener algún modelo de TM para poder realizar predicciones sobre la variación de la demanda a medio y largo plazo. Con esta información el operador podría reconfigurar la red para controlar picos de tráfico y de esta manera evitar la congestión de la red.

- Detección de anomalías: existen diferentes anomalías que un operador de red puede o está interesado en detectar. Por ejemplo, la probabilidad de que un grupo de usuarios --o únicamente uno-- muestre un cambio de comportamiento, por ejemplo, incrementando su demanda de tráfico. En este caso, el modelo debería ser capaz de señalar que algún problema está ocurriendo en la red y ésta daría una respuesta para volver a una situación de normalidad. Otra posible anomalía podría ser el hecho de que un usuario malicioso o incluso un error de configuración en la red efectuara una gran cantidad de peticiones a uno de los nodos. Este fenómeno es el llamado ataque de denegación de servicio (DoS, *Denial of Service*), y en el caso de que fuera provocado intencionadamente, es utilizado para bloquear algún servicio o host por sobrecarga de tráfico de entrada. En este caso, el modelo debería mostrar que algún hecho está provocando un aumento repentino del volumen de tráfico entre nodos de la red, fuera de lo habitual o esperado en condiciones normales. Respecto al balanceo de carga, la detección de estas anomalías pueden ayudar a reconfigurar las rutas de la red y sus cargas.

- Predicción de TMs: uno de los ámbitos más importantes de investigación es encontrar un modelo que proporcione una predicción fiable de las futuras TMs, para poder compararlas con las reales y de esta forma ser capaz de detectar las anomalías. El hecho de adelantarse a los cambios y de actuar antes de que sucedan significará un gran avance. Desafortunadamente no existen sólidos conocimientos sobre la manera de obtener un modelo fiable, a pesar de que muchos trabajos de investigación ya están centrados en este ámbito [16, 17].

1.3. Obtención de las matrices de tráfico

Como se ha podido ver hasta ahora, las matrices de tráfico son una medida de gran importancia para la ingeniería de tráfico y la planificación y dimensionado de redes; sin embargo, aparecen problemas para obtenerlas de una manera sencilla. El primer obstáculo estriba en cómo efectuar las medidas en tiempo real, y es aquí donde aparecen dos técnicas distintas pero con la misma finalidad: NetFlow y los contadores SNMP.

1.3.1. NetFlow

NetFlow [9, 13] es el nombre que se le da tanto a una aplicación como a un protocolo de red desarrollados por Cisco Systems para recolectar información del tráfico IP. Otros fabricantes ofrecen productos similares con distinto nombre, como por ejemplo Jflow/cflowd para redes Juniper o Rflow para Ericsson. Una de sus características es que permite medir flujos y volúmenes de tráfico conmutados por un router. Los routers que tienen activada la aplicación NetFlow generan registros de NetFlow (NetFlow records); estos se exportan del router en paquetes UDP (Fig. 1.4) o SCTP (Stream Control Transmission Protocol) y son recogidos utilizando un colector de NetFlow.

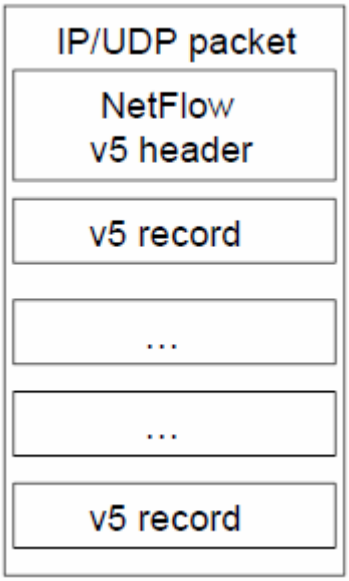


Fig. 1.4 Paquete v5 de NetFlow

El principal inconveniente de utilizar este protocolo es que cuando un nodo recibe una gran cantidad de flujos, NetFlow requiere una alta capacidad de CPU para procesar la información recibida, en detrimento de las tareas de encaminamiento, por lo que es complicado su uso en redes IP troncales, MANs o WANs (redes de área metropolitana o extensas), como las que se han utilizado en este estudio. Una posibilidad para reducir esta carga es Sampled NetFlow, que consiste en muestrear sólo una fracción de los paquetes, pero esto puede reducir la exactitud de las medidas. Otra razón es la dificultad para realizar medidas en tiempo real: existen problemas de sincronía al detectar un mismo flujo en diferentes routers al mismo tiempo pero no de igual manera, ya que quizá en unos sí se muestrea y en otros no, y no necesariamente en el mismo instante.

1.3.1.1. Arquitectura

Un sistema NetFlow está formado por tres componentes (Fig. 1.5):

- Exportador: analiza y recolecta los flujos de datos IP que entran al router o switch y prepara la información para ser exportada.
- Colector: captura los datos exportados por múltiples routers y filtra y agrega la información de acuerdo a los criterios establecidos por el operador de red. También puede integrar información externa a los registros (por ejemplo, añadiendo atributos BGP).
- Analizador: es un software de monitorización que analiza los datos que provienen del colector, pudiendo realizar informes sobre el tráfico de red.

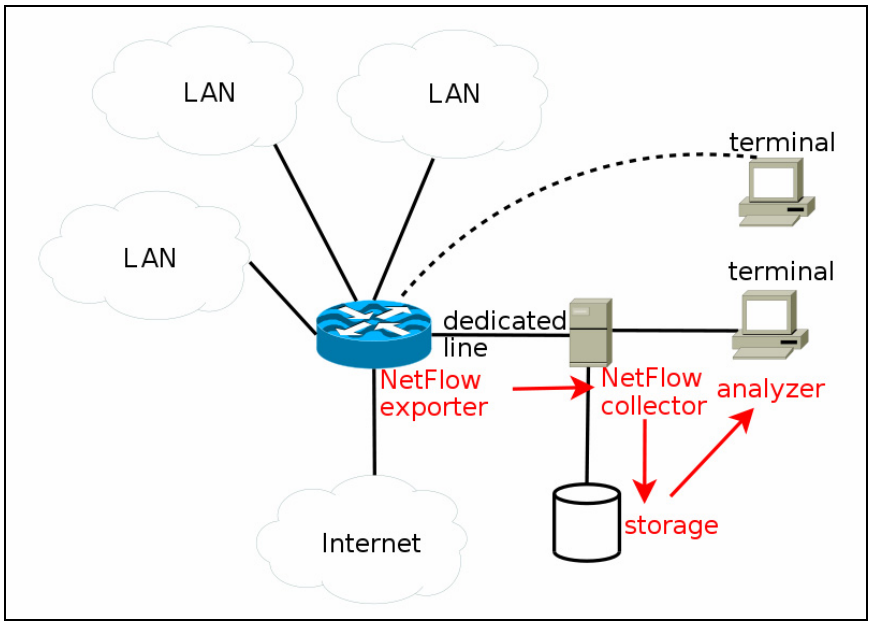


Fig. 1.5 Arquitectura NetFlow

1.3.1.2. Formato de los registros

Los flujos de un router se exportan en un formato para el que existen diferentes versiones, pero en este estudio se ha trabajado con registros pertenecientes a la versión 5 de NetFlow. Un flujo es una colección de campos clave (Fig. 1.6) e información adicional.

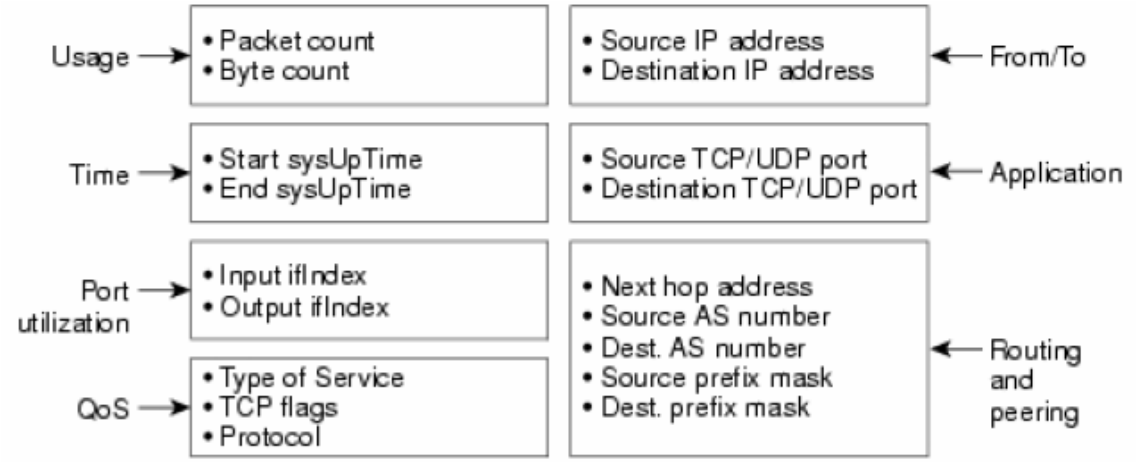


Fig. 1.6 Registro NetFlow donde aparecen todos los campos clave

1.3.2. SNMP (Simple Network Management Protocol)

SNMP es un protocolo de red, transportado habitualmente sobre UDP, que se utiliza para la monitorización de los equipos de red. Guarda la información de gestión en variables almacenadas en bases de datos en los propios equipos que tienen activado este protocolo. Estas variables pueden ser luego consultadas (y en ocasiones configuradas) por las aplicaciones de gestión. En un uso típico de SNMP, uno o más ordenadores llamados *managers* tienen la tarea de monitorizar o gestionar a un grupo de hosts o dispositivos en una red. Cada sistema gestionado ejecuta en todo momento un programa llamado *agente* que informa mediante SNMP al manager.

A diferencia de NetFlow, las consultas a los routers mediante este protocolo no afectan a las tareas de encaminamiento debido a que se limita a escribir o leer de bases de datos llamadas MIBs (Management Information Bases), operaciones que requieren un coste computacional muy bajo. En estas bases de datos se pueden encontrar, por ejemplo, la cantidad de datos transmitidos por un enlace (sin discriminar por destino final, es decir, sin almacenar la ruta) a una frecuencia típica de cinco minutos. El problema es que los datos que se obtienen son el valor agregado de todos los flujos que circulan a través del enlace, por lo que es necesario realizar un procedimiento de inferencia para diferenciar los distintos flujos (que son los componentes que sumamos para obtener una matriz de tráfico).

En las figuras 1.7 y 1.8 se pueden ver dos capturas de los contadores SNMP de RedIRIS extraídas de su Weathermap[3].

[3] http://www.rediris.es/conectividad/weathermap

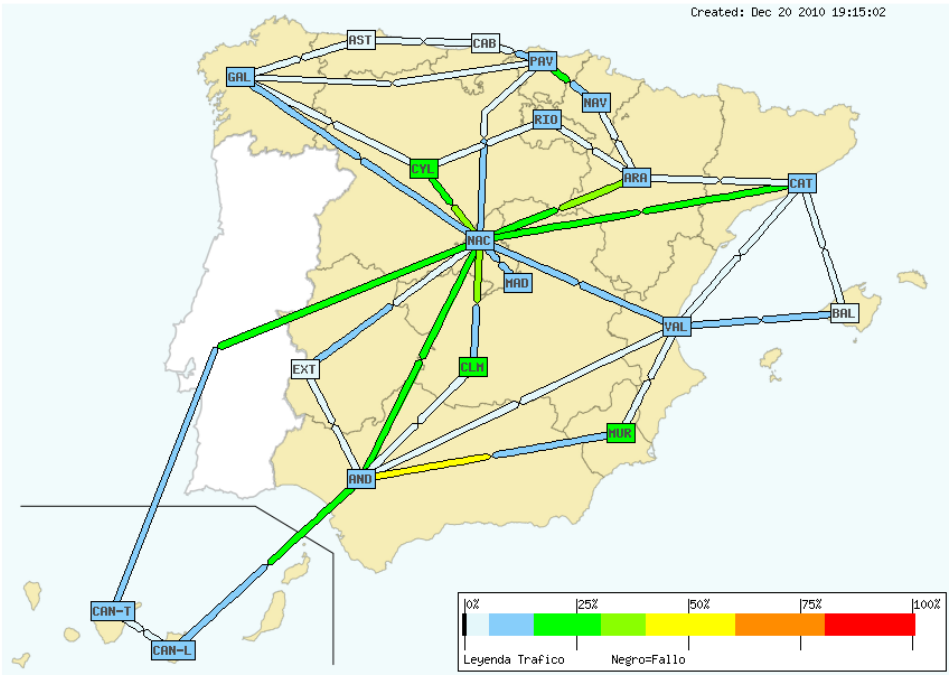


Fig. 1.7 Mapa global de RedIRIS

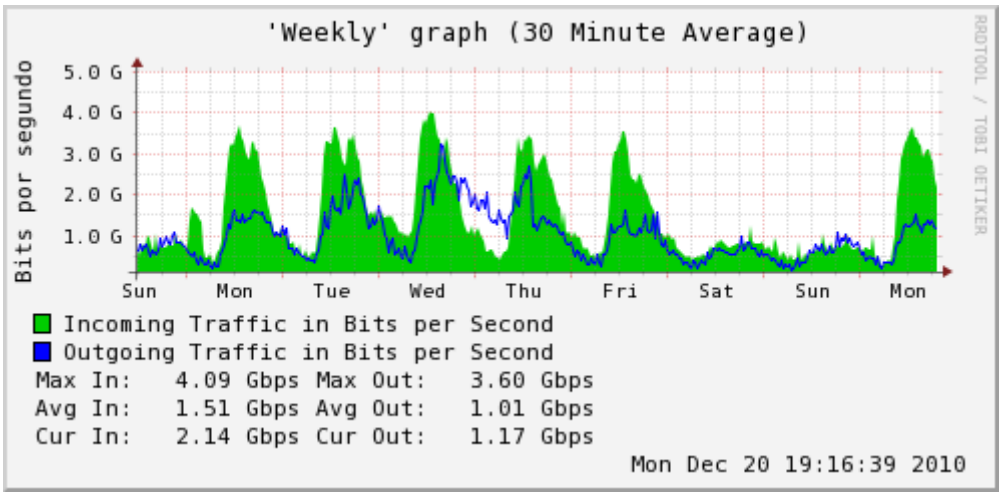


Fig. 1.8 Estadísticas del enlace Nacional – Madrid de RedIRIS

El segundo problema proviene del hecho de que los proveedores de servicios de Internet no están dispuestos a publicar estos datos, y es comprensible debido a la competencia entre empresas. Este es uno de los motivos principales en la lentitud de la investigación en este campo. A los investigadores en algunas ocasiones se les permite la captura de las muestras en tiempo real para las propias organizaciones, pero es muy difícil conseguir conjuntos de datos de larga duración. Se ha de agradecer a Zhang y Uhlig [17, 22] la publicación de dos conjuntos de datos de GÉANT y Abilene correspondientes a 2004 y 2005, que son los utilizados en nuestros

experimentos y representan el punto de partida para la investigación del modelado de matrices de tráfico.

1.3.3. Inferencia de matrices

Con SNMP se obtiene la información sobre el tráfico para cada uno de los enlaces, pero a diferencia de NetFlow, con el que se obtienen datos de cada flujo, con SNMP no se obtiene toda la información, y en consecuencia para completar las matrices se ha de realizar algún tipo de inferencia. Esta falta de datos reales se puede solucionar haciendo una estimación de los datos no disponibles. Para esta estimación se ha de utilizar la siguiente expresión:

$$y = Ax \qquad \textbf{(1.1)}$$

donde *Y* representa el vector de contadores SNMP de enlaces (de longitud *K*, donde K representa la cantidad de enlaces en la red). *A* es la matriz de encaminamiento (a_{ij}=1, si el enlace *i* pertenece a la ruta asociada al par *j* origen – destino), con dimensiones *K* x N^2, siendo N el número de nodos, y *x* es la matriz de tráfico (escrita como un vector N^2 x *1*, x_{ij} es el valor del tráfico asociado con el par *i-j* origen - destino). En el apartado 2.2.4 se amplia la información sobre la matriz de encaminamiento.

Desafortunadamente, como *K* es un valor mucho más pequeño que N^2 (Abilene cuenta con 12 nodos y 42 enlaces, por lo que *K* = 42 y N^2 =144) deducir una matriz de tráfico a partir de las medidas SNMP se convierte en un problema inverso irresoluble con infinitas soluciones, y por tanto se necesitan datos adicionales para obtener una solución. Uno de los modelos que aportan información extra es el Modelo de Gravedad (*Gravity Model*).

1.3.4. Modelo de Gravedad

El Modelo de Gravedad para matrices de tráfico [23] está basado en la ley de gravedad de Newton, que indica que la fuerza de atracción entre dos cuerpos es directamente proporcional al producto de sus masas y una constante gravitacional, e inversamente proporcional al cuadrado de la distancia existente entre los dos cuerpos. Esta idea también se utiliza en otros ámbitos, como pueden ser ciencias sociales y el transporte de mercancías, donde se encuentran modelos de gravedad para las transacciones económicas o incluso en flujos migratorios.

En todos estos casos, la formulación del modelo puede llevar a una buena analogía entre el parámetro original por masa y distancia y unos parámetros adaptados para cada interpretación de la ley de la gravedad. El modelo de gravedad depende del producto de las masas (población, terminales) de dos

cuerpos (ciudades, nodos), un valor constante (constante gravitacional universal en la formulación original) y la distancia (correlación, adyacencia). En el caso de migraciones de población, la tendencia es de migrar a grandes ciudades (cuanto mayor es la masa, más fuerte es la fuerza de atracción), mientras que en el caso de la economía, los mercados económicos gigantes concentran la mayor parte de los negocios.

En nuestro caso, el modelo de gravedad asume la hipótesis de que el tráfico que circula entre dos ciudades con gran población será mayor que el que circula entre dos ciudades con una población menor. Como la fuerza gravitacional, la cantidad de información que viaja entre Madrid y Barcelona será elevada, debido a que existe más "población" (terminales) que utilizan los servicios de un operador. Por contra, la cantidad de datos que circulan entre Castelldefels y Barcelona debería de ser menor que la transportada entre Madrid y Barcelona, pero las dos cantidades deberían depender, de alguna manera, de la población de Barcelona.

Para calcular el modelo de gravedad para una matriz de tráfico no se considerará la distancia física: el parámetro original de la ley de la gravedad referido a la distancia no es tan evidente en este modelo, ya que los algoritmos de encaminamiento en las redes pueden encaminar los paquetes a través de caminos alternativos, por tanto esta distancia no puede hacer referencia a la distancia física.

Es por este motivo que se utilizarán volúmenes de tráfico de ingreso-egreso en lugar de utilizar directamente la población. Con ello se obtendrán 2N parámetros (N por la fracción de tráfico total generado para cada uno de los nodos y una N más para la fracción del tráfico recibido total por cada nodo). Esta cantidad de información permite obtener las medidas reales a partir de los contadores SNMP y además estimar la información restante.

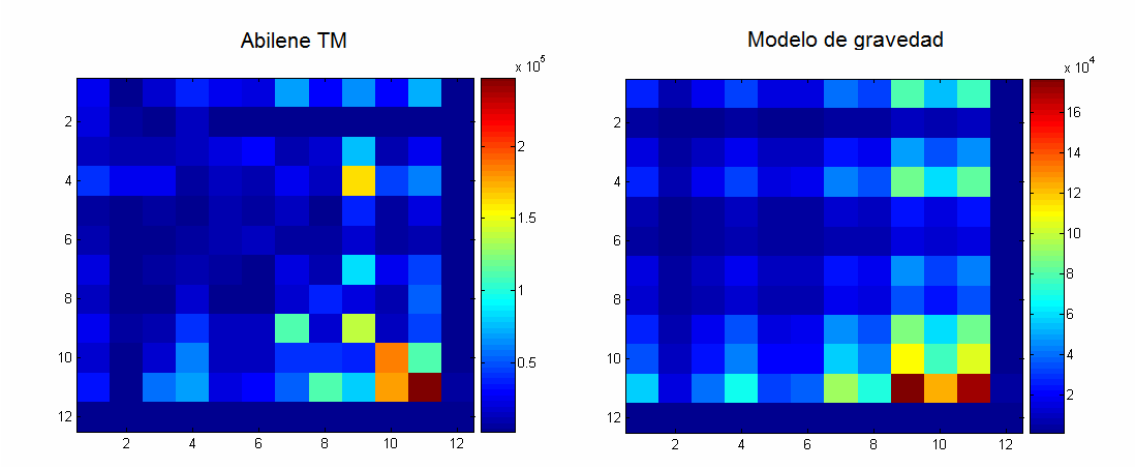


Fig. 1.9 Modelo de gravedad (GM). Izquierda: matriz de tráfico de Abilene (06/09/2004 16:45). Derecha: Modelo de gravedad de la matriz previa

Cuando se obtiene la matriz de gravedad, el volumen total de tráfico se conserva; por tanto, se puede afirmar que la energía se conserva. La ecuación es la siguiente:

$$GM = T_i \times T_o^T \times TráficoTotal \quad \textbf{(1.2)}$$

donde T_i es el vector de probabilidades de entrada y T_o^T es el vector de probabilidades de salida de tráfico por cada nodo.

En la matriz de la figura 1.9 el tráfico total es de 3.5904e+06 (359,04 MB. Las unidades de las matrices de Abilene son 100 bytes / 5 min.); el volumen se mantiene tras aplicar el modelo de gravedad:

T_i =	T_o =	TráficoTotal =
0.0633	0.1110	3.5904e+06
0.0164	0.0131	
0.0440	0.0670	
0.0775	0.1179	
0.0371	0.0331	
0.0410	0.0224	
0.1039	0.0636	
0.0801	0.0496	
0.2005	0.1215	
0.1393	0.1514	
0.1921	0.2475	
0.0048	0.0019	

Una manera de interpretar el modelo de gravedad es como la aproximación de baja dimensionalidad (dimensión 1, ya que sólo existe una fila y una columna independientes) de una señal original de dimensionalidad mucho más alta.

Sin embargo, el modelo de gravedad no es necesariamente una buena representación de la matriz real, ya que la matriz de tráfico debe cumplir siempre la ecuación (1.1), que relaciona la carga de cada enlace con la matriz de tráfico y con la de encaminamiento, y el modelo de gravedad no necesariamente lo verifica. La solución a este problema es el método Tomogravedad.

1.3.4.1. Tomogravedad y programación cuadrática

En este estudio se comprueba el método Tomogravedad desarrollado por Roughan *et al*. aplicando el código publicado en [23] para calcular la matriz de tráfico a partir de los datos de los enlaces y la información de encaminamiento [11], obteniendo una matriz de tráfico estimada (TM'). La estimación resultante es la denominada *solución del modelo de gravedad,* pero tal como se ha comentado previamente, esta representación no verifica la ecuación 1.1, por lo que posteriormente se refina la solución del modelo de gravedad usando una solución de mínimos cuadráticos que minimiza la distancia euclidiana a la

solución del modelo de gravedad sujeta a las restricciones de la tomogravedad (Fig. 1.10).

También se utilizan soluciones ponderadas de mínimos cuadráticos, en las que la proyección al subespacio no es ortogonal, sino ponderada por una función del tamaño de los elementos de la matriz de tráfico estimada. Este vector ponderado es *w* y la ponderación (peso, en términos de programación cuadrática) puede ser constante, linealmente proporcional al modelo de gravedad o proporcional a la raíz del modelo de gravedad.

La figura 1.10 ilustra la solución propuesta que se realiza en un caso simple con dos incógnitas (tráfico desconocido en dos rutas), y una restricción (carga conocida de un enlace). La restricción especifica un subespacio del parámetro espacio (en este caso sólo una línea), mientras que el modelo de gravedad es una solución particular. La solución de mínimos cuadráticos simple es una proyección ortogonal de la solución del modelo de gravedad en el subespacio. La solución de mínimos cuadráticos ponderados otorga diferente ponderación a las diferentes incógnitas de la solución. En la figura el punto muestra la solución del modelo de gravedad, y la línea discontinua el subespacio especificado por una ecuación. La solución de mínimos cuadráticos es el punto que cumple la ecuación, que es la que está más cerca (en geometría ecuclidiana) a la recta de soluciones de y=Ax. La solución ponderada de mínimos cuadráticos otorga un peso diferente a diferentes incógnitas.

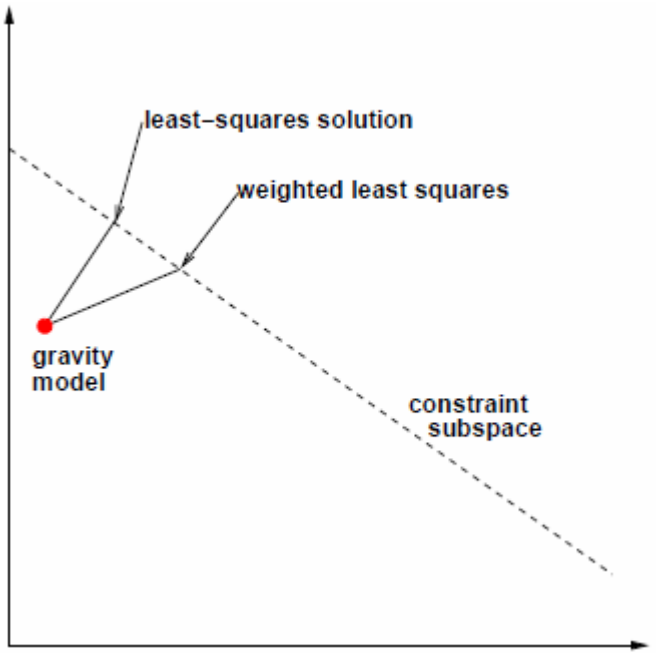


Fig. 1.10 Solución de mínimos cuadráticos

Las restricciones pueden no ser correctas por culpa de errores y ruido en los datos transportados en el enlace o a posibles cambios de rutas que no se capturan en la información de la topología. La técnica estándar para tratar con programaciones cuadráticas mal formuladas es utilizar la denominada

Descomposición del Valor Singular (SVD, por sus siglas en inglés) de la matriz de encaminamiento *A* para calcular su seudoinversa. La solución resultante es la más cercana a la solución inicial entre todas las soluciones que minimizan la discrepancia de las restricciones tomográficas. El algoritmo utilizado para ello es el publicado en [23].

Para finalizar, se ha de comentar que los autores declaran [23] que el método proporciona un ±20% de error (medido como error cuadrático medio, MSE, entre la matriz de tráfico original y la estimada por el procedimiento de tomogravedad), utilizando la ponderación proporcional a la raíz del modelo de gravedad.

CAPÍTULO 2. ANÁLISIS DE MATRICES DE TRÁFICO DE ABILENE Y GÉANT

2.1. Descripción de los conjuntos de datos

Para la realización de los análisis de matrices de tráfico, en este trabajo se han utilizado datos provenientes de las redes académicas Abilene (estadounidense [1]) y GÉANT (europea [7]). Se ha de tener en cuenta que ambas organizaciones facilitan la mínima información posible sobre su estructura, y el principal motivo es para prevenir que usuarios maliciosos tengan el conocimiento suficiente como para causar un daño en sus redes.

Los datos proporcionados por Abilene y GÉANT están incompletos, en algunas ocasiones tan solo falta una matriz pero en otras puede haber vacíos de horas o incluso días. En el caso de Abilene, las matrices facilitadas suelen ser grupos de varios días. En GÉANT las matrices agrupadas en días seguidos son más complejas de obtener. En el momento de realizar este trabajo, ésta ha sido una de las circunstancias que se han tenido en cuenta, ya que el hecho de no tener datos continuos, sobre todo en el caso de GÉANT, ha supuesto descartar grupos de datos que no daban continuidad a los análisis, y por tanto había una cierta incongruencia en los resultados. En el caso de Abilene, al ser grupos de datos más continuos, a la hora de realizar ciertos cálculos, ha sido posible visualizar los ciclos diarios e incluso semanales, hecho que con GÉANT, como se ha comentado debido a su frecuencia de obtención de matrices, ha sido más complejo.

2.1.1. Abilene

Abilene decidió publicar sus datos, el mapa de la red y los enlaces de cada nodo a los puntos de presencia (PoPs), pero siempre protegiendo esta información porque es obsoleta o con desviaciones temporales, ya que los conjuntos de datos utilizados pertenecen supuestamente al periodo comprendido entre los meses de marzo y septiembre de 2004, pero no ha habido ninguna forma de poder asegurarlo. En total se disponen de 48.096 matrices de tráfico, tal y como se puede ver en la tabla 2.1.

Tabla 2.1. *Data set* de Abilene

Mes	Marzo	Abril	Mayo	Junio	Julio	Agosto	Sept.	**Total**
TMs	4.032	6.048	8.928	8.640	8.928	8.640	2.880	**48.096**

Las muestras de Abilene están tomadas con una frecuencia de cinco minutos, haciendo un total de doce por hora.

2.1.2. GÉANT

GÉANT, aparte de publicar matrices de tráfico antiguas, oculta alguna información relativa a los nodos: todos los datos están anonimizados y no es posible saber qué nodos corresponden a cada PoP. Además se pueden encontrar muestras que no concuerdan en tiempo con los nombres de las matrices; por ejemplo, *IntraTM-2005-03-05-07-30* no han de ser necesariamente datos del 5 de marzo de 2005 a las 7:30 horas.

En el caso de GÉANT sólo está disponible una matriz cada quince minutos, resultando en cuatro por hora y haciendo un total de 10.772 matrices de tráfico, correspondientes al periodo enero – abril de 2005.

Tabla 2.2. *Data set* de GÉANT

Mes	Enero	Febrero	Marzo	Abril	**Total**
TMs	2.941	2.310	2.765	2.756	**10.772**

2.2. Utilización de los datos

Una vez obtenidos los datos, se tuvo que estudiar su topología física para poder trabajar con las matrices de tráfico. En un principio se utilizaron los datos de Abilene, que al ser más continuos permitieron un tratamiento más simple que los de GÉANT. El proyecto TOTEM [21] podría haber permitido mostrar las topologías de ambas redes e incorporar TMs para visualizar su comportamiento, pero no se ha empleado. Para realizar los análisis numéricos se ha utilizado Matlab [10], ya que se dispone de código y funciones que hacen que el tratamiento de los datos sea relativamente simple.

2.2.1. Análisis de los datos

Una vez obtenidos los *data sets* es necesario analizarlos para trabajar con las matrices en Matlab. Las matrices de tráfico están en formato XML, con o sin etiquetas, dependiendo de si son de Abilene o GÉANT.

```
<?xml version="1.0" encoding="ISO-8859-1"
standalone="yes"?>
<TrafficMatrixFile […]>
<IntraTM ASID="11537">
<src id="ATLA-M5">
<dst id="ATLA-M5">26.6666666666667</dst>
<dst id="ATLAng">522.208</dst>
<dst id="CHINng">1641.33866666667</dst>
<dst id="DNVRng">335.728</dst>
<dst id="HSTNng">413.032</dst>
<dst id="IPLSng">489.874666666667</dst>
<dst id="KSCYng">365.077333333333</dst>
<dst id="LOSAng">817.869333333333</dst>
<dst id="NYCMng">452.061333333333</dst>
<dst id="SNVAng">747.405333333333</dst>
<dst id="STTLng">388.317333333333</dst>
<dst id="WASHng">3141.64</dst>
</src>
[…]
</IntraTM>
</TrafficMatrixFile>
```

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<TrafficMatrixFile […]>
<info> […] </info>
<IntraTM ASID="20965">
<src id="12">
<dst id="12">302691.3422</dst>
<dst id="13">30623.7689</dst>
<dst id="19">9156.3289</dst>
<dst id="23">2726.5867</dst>
<dst id="8">5659.2267</dst>
<dst id="18">4138.5511</dst>
<dst id="4">152568.9067</dst>
<dst id="1">8988.0622</dst>
<dst id="5">4168.9600</dst>
<dst id="3">15538.8000</dst>
<dst id="22">17130.4267</dst>
<dst id="7">2514.1689</dst>
<dst id="2">101214.7556</dst>
<dst id="6">18.0089</dst>
<dst id="16">8901.9911</dst>
<dst id="14">434.1511</dst>
<dst id="20">10964.5867</dst>
<dst id="11">4511.0489</dst>
<dst id="9">4441.8667</dst>
<dst id="17">86795.6356</dst>
<dst id="21">13459.1644</dst>
<dst id="15">3841.1556</dst>
</src>
[…]
</IntraTM>
</TrafficMatrixFile>
```

Fig. 2.1 Dos ejemplos de TMs en formato XML de Abilene (izquierda) y GÉANT (derecha)

2.2.2. Matrices de tráfico en Matlab

Matlab es una aplicación matemática desarrollada por Mathworks. Permite analizar estadísticamente las matrices y ofrece una gran cantidad de funciones relacionadas con diversos campos matemáticos.

Una característica que aporta Matlab es la posibilidad de visualizar las TMs. Con este propósito, se ha desarrollado código para pasar los datos de los archivos XML a Matlab y poderlos representar. Si se trabaja sobre las matrices directamente, esto es, con los valores numéricos, resulta muy complicado apreciar las diferencias y detectar algún tipo de patrón o anomalía. Aunque Matlab ofrece distintas formas para representar matrices de datos, en este trabajo sólo se ha utilizado la forma de representación *imagesc*. En la figura 2.2 se puede ver el tipo de gráfico creado por Matlab con la función *imagesc*. La matriz representada pertenece a Abilene, es el fichero XML “TM-2004-09-06-1645.xml” (6 de septiembre de 2004 a las 16:45 horas).

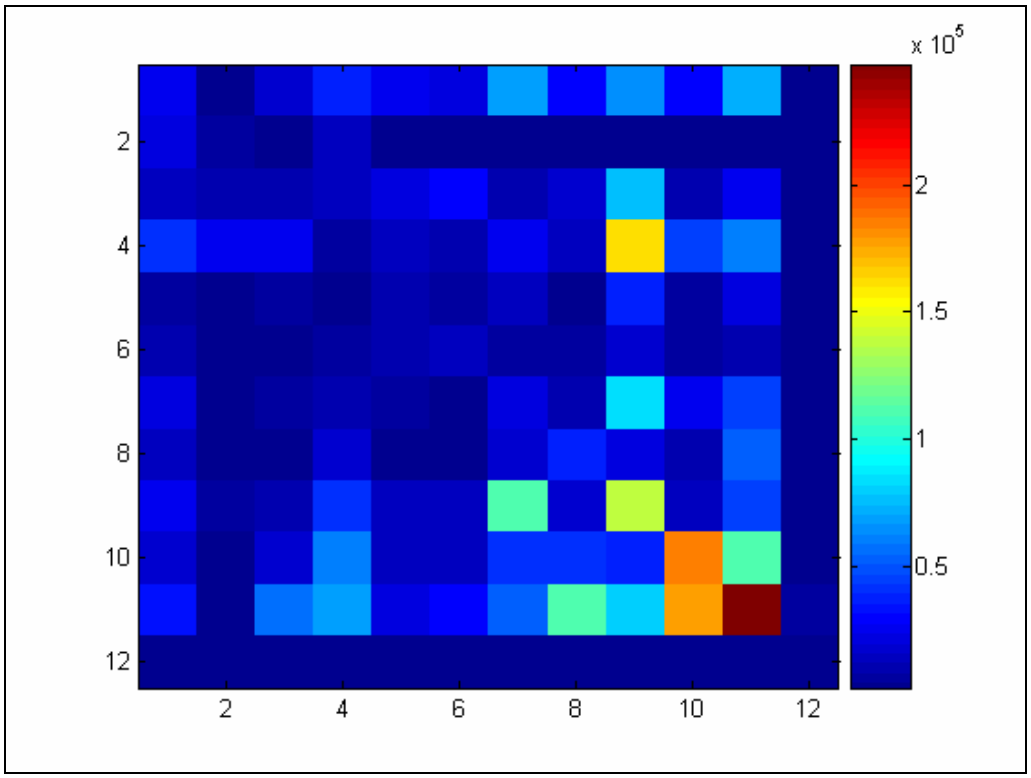


Fig. 2.2 Representación de la matriz TM-2004-09-06-1645 de Abilene mediante *imagesc* de Matlab

Este tipo de gráfico permite obtener una representación visual de los parámetros de la matriz de tráfico. La asociación gráfica de los colores está relacionada al valor numérico de cada posición de la matriz. Por tanto, en el caso de visualizar un bloque de matrices consecutivas, se podrían detectar patrones de comportamiento o anomalías de una forma muy sencilla si los colores, y por tanto los valores relacionados a estos, cambiasen. Se ha de comentar que el etiquetado de cada fila o columna (correspondiente a un PoP de origen o destino, respectivamente) es arbitrario.

2.2.3. Matriz de adyacencia

Otro tipo de matrices que se presentan en el estudio son las matrices de adyacencia. Estas matrices indican qué nodos tienen enlaces directos entre sí. Siempre que la relación del enlace de un nodo y otro sea directa, estos tendrán un valor igual a 1 en la posición de la matriz que relacione estos nodos.

A modo de ejemplo, si se dispone de una matriz M, la matriz de adyacencia se representaría de la siguiente manera: $b_{ij} = 1$ cuando existe un enlace entre los nodos i y j, $b_{ij} = 0$ si no existe enlace directo entre los nodos i y j. De esta manera la matriz binaria indica la relación de vecindad entre los nodos de la red o grafo.

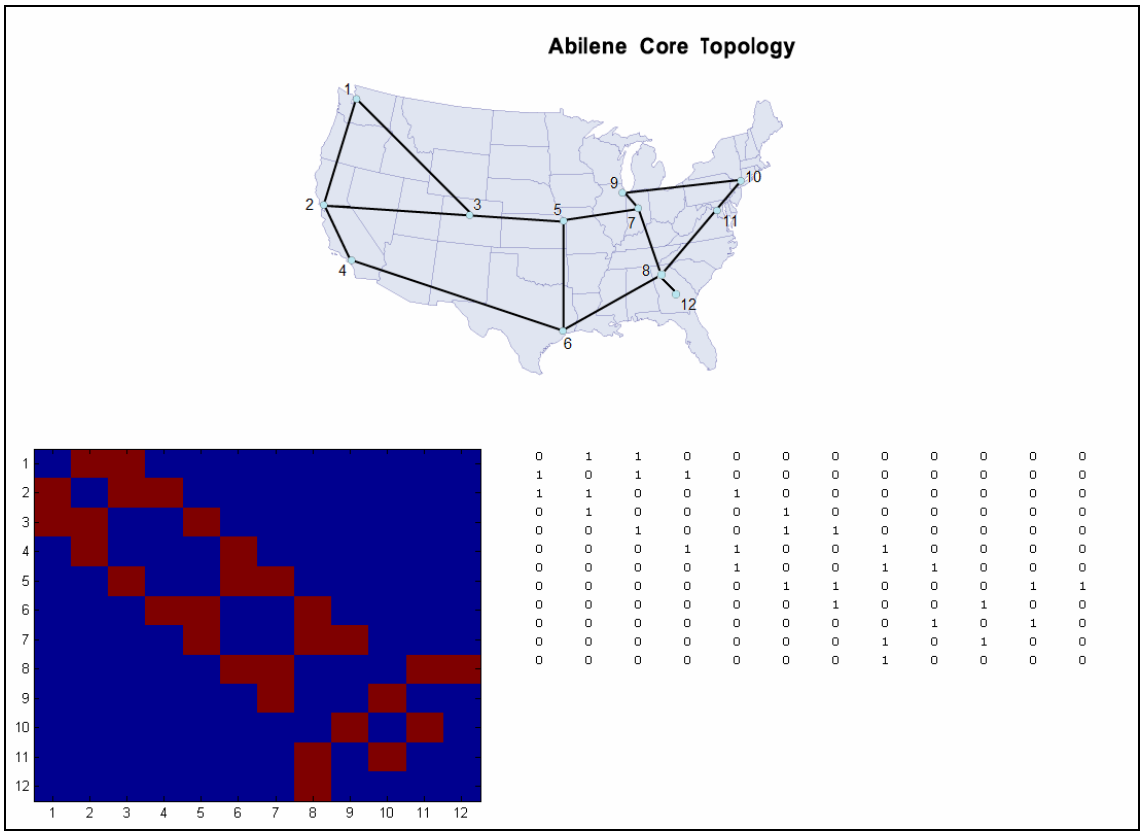


Fig. 2.3 Relación entre la red Abilene (grafo) y su matriz de adyacencia

En la figura 2.3 se puede visualizar la representación de la matriz de adyacencia de la red Abilene y sus valores. La analogía es visible: los píxeles rojos son los que tienen valor 1 y los azules 0. Cada valor [i,j] representa una combinación de los dos nodos: el primer 0 representa la adyacencia de 1 (Seattle) a 1 (Seattle); es cero porque él mismo no puede ser su propio vecino. Si se observa la primera línea, aparecen los nodos vecinos de 1 en el siguiente orden: 2 (Sunnyvale) y 3 (Denver). Se puede apreciar que en este caso la diagonal de la matriz de adyacencia es cero (la diagonal representa las adyacencias de cada nodo con él mismo) aunque en el caso del estudio también se considera el tráfico que va de un nodo hacia él mismo, y que por tanto deberá ser añadida a la matriz identidad.

2.2.4. Matriz de encaminamiento

La matriz de encaminamiento representa la relación de enlaces existentes en una red: a_{ij}=1, si el enlace *i* pertenece a la ruta asociada al par *j* origen – destino, con dimensiones *K* x N^2, siendo K la cantidad de enlaces en la red y N el número de nodos. Esta matriz es necesaria para poder aplicar el modelo de gravedad y posteriormente calcular la matriz estimada mediante el código publicado por Roughan *et al.* en [23]. En la figura 2.4 se puede ver la representación de la versión reducida de la matriz de encaminamiento de Abilene (K = 42, N=12, N^2 =144). En esta representación el píxel es negro cuando hay un enlace, y blanco cuando no lo hay. Es “reducida” en el aspecto

de representar sólo los 42 enlaces presentes en la red, y no los Nx(N-1) posibles enlaces totales.

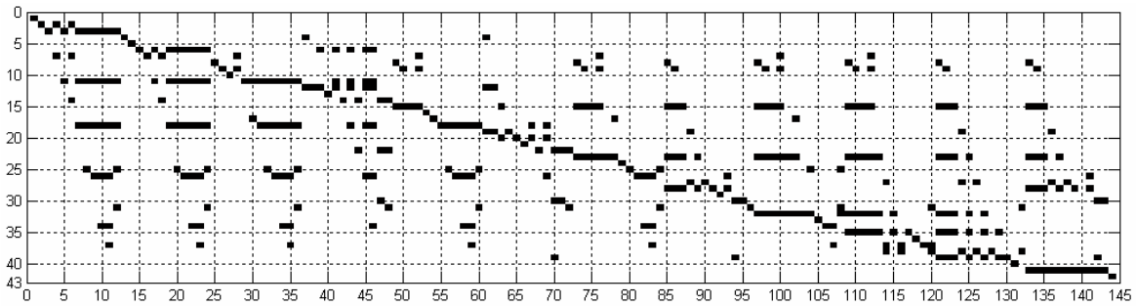


Fig. 2.4 Matriz de encaminamiento reducida de Abilene

2.3. Aplicación del método de Tomogravedad

En este estudio se comprueba el método de Tomogravedad desarrollado por Roughan *et al*. en [23] aplicando el código de la figura 2.5 para calcular la matriz de tráfico a partir de los datos de los enlaces. En Abilene, este cálculo se ha realizado utilizando tres ponderaciones (w) distintas, tal y como sugieren los autores del código: constante (w=ones), linealmente proporcional al modelo de gravedad (w=gravity_model) y proporcional a la raíz del modelo de gravedad (w=sqrt(gravity_model)). Para GÉANT únicamente se utilizó el vector ponderado proporcional a la raíz del modelo de gravedad, ya que tras los análisis de Abilene se verificó que era el que mejores resultados ofrecía. Una vez obtenida la matriz de tráfico estimada (TM'), se calcula el error medio respecto a la TM original para conocer la diferencia de potencia entre ésta y la estimada.

```
% weighted least-squares estimate of the TM
% Input:
% A matrix A in constraints x=A*t
% x vector x in constraints x=A*t
% tg initial gravity model solution
% w weight vector
% Output:
% t estimated traffic matrix (as a vector)
% that minimizes |(t-tg)./w|
% among all t's that minimize |A*t-x|
function [t] = wlse(A,x,tg,w)
% equivalently transform x=A*t into
% xw=Aw*tw, where tw=(t-tg)./w
xw = x - A*tg;
[r, c] = size(A);
Aw = A .* repmat(w', r, 1);
% solve tw=Aw*tw by computing the pseudo-
% inverse of matrix Aw (through svd)
tw = pinv(full(Aw)) * xw;
% transform tw back to t
t = tg + w .* tw;
```

Fig. 2.5 Código de M. Roughan para calcular la TM estimada (TM')

El algoritmo diseñado por M. Roughan para calcular la matriz de tráfico estimada mediante el código de la figura 2.5 es el siguiente:

- Aplicar el modelo de gravedad generalizado a la matriz de tráfico original para obtener una TM de enlace a enlace.
- Transformar la TM anterior en una matriz de tráfico PoP a PoP (Punto de Presencia), y utilizarla como la solución inicial del modelo de gravedad, *tg*.
- Reducir la complejidad del problema de tomografía quitando los PoPs sin demanda de tráfico.
- Aplicar SVD (Descomposición del Valor Singular) para resolver la ecuación cuadrática y encontrar una solución que minimice su distancia a *tg* respetando las restricciones del modelo tomográfico.
- Reemplazar los valores negativos con cero y realizar IPF (Iterative proportional fitting) para obtener una solución no negativa que cumpla las restricciones. IPF es un algoritmo iterativo para estimar los valores de una matriz, cuyos elementos cumplen ciertas condiciones, sin perder la proporcionalidad entre los elementos de una matriz inicial utilizada como primer paso de la iteración.

2.3.1. Procedimiento

El procedimiento para hallar la matriz de tráfico estimada en Matlab es el siguiente: para aplicar el código de M. Roughan (de aquí en adelante *wlse*) es necesario haber calculado mediante el código del anexo A.II, new_routing_matrix (A), link_loads (x), la solución del modelo de gravedad (tg) (Fig. 2.6) y escoger un vector de pesos (w).

Una vez obtenido new_routing_matrix (A) y link_loads (x), ya se puede aplicar el código de M. Roughan para conseguir la matriz estimada (TM'), tal como se muestra en la figura 2.7.

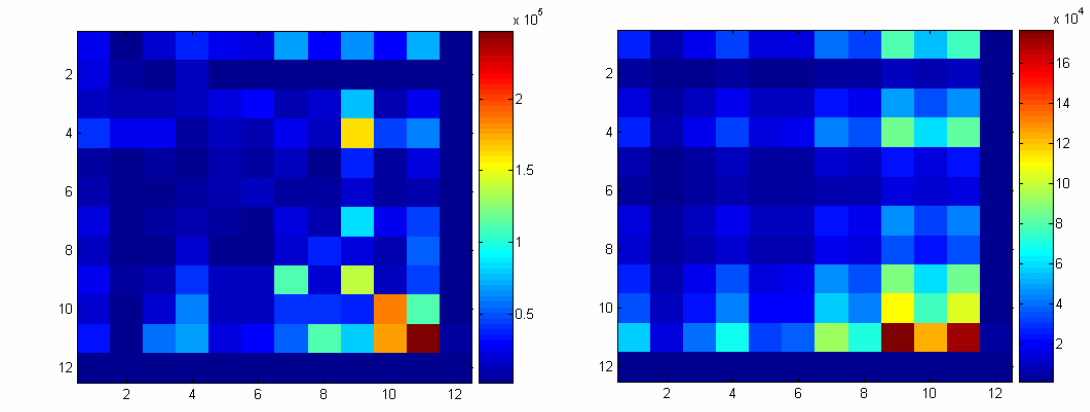


Fig. 2.6 Izquierda, TM original (2004-09-06-1645.xml). Derecha, *tg*, solución del modelo de gravedad de la TM original

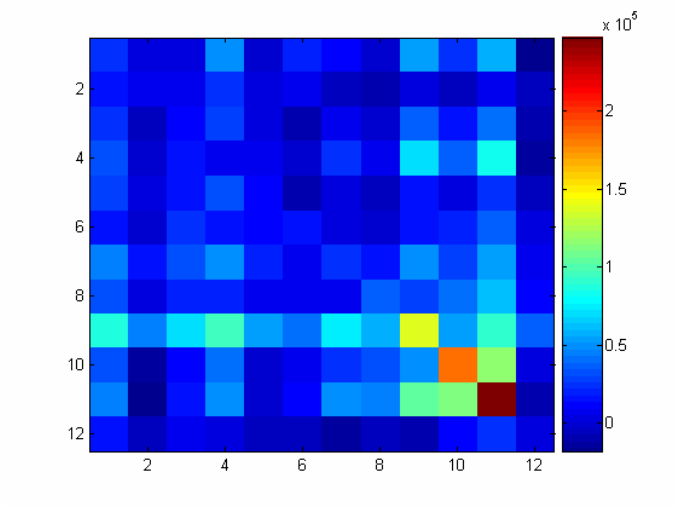


Fig. 2.7 TM estimada mediante el código de M. Roughan

Una vez obtenida la matriz de tráfico estimada, se calcula el error entre ésta y la original. En este caso concreto, para la matriz 2004-09-06-1645.xml, la matriz estimada difiere solamente en un 4,38% de la potencia de la matriz original, utilizando un vector de pesos (w) constante (ver anexo A.II).

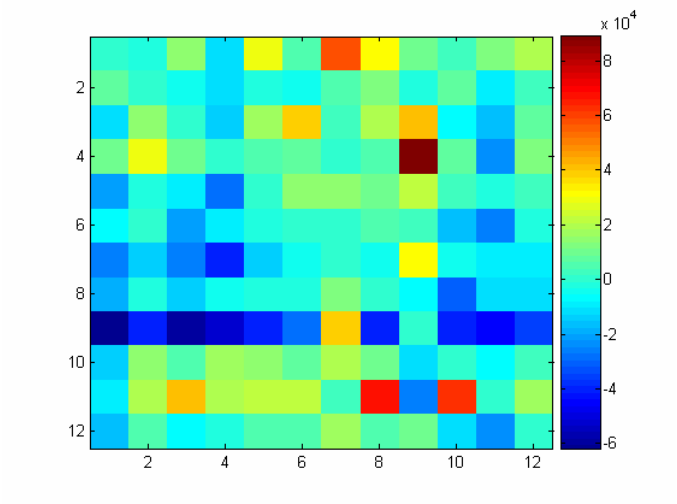


Fig. 2.8 Representación del error entre TM original (2004-09-06-1645.xml) y la estimada (TM')

2.3.2. Resultados

En el punto A.II. del anexo aparecen recogidos los cálculos realizados en el análisis de las matrices de tráfico de Abilene y GÉANT.

2.3.2.1. Abilene

Se analizó el mes 05 de 2004 por ser uno de los más completos en el *data set*, no habiendo vacíos temporales significativos que pudieran influir en los cálculos realizados. En total se trataron 8.928 matrices de tráfico. Los resultados aparecen recogidos en la tabla 2.3.

Tabla 2.3. Error entre la TM original y la estimada mediante Tomogravedad

Ponderación	Error MSE
w=constante	32,56%
w=gravity_model	41,36%
W=sqrt(gravity_model)	31,94%

2.3.2.2. GÉANT

Se analizó el mes 01 de 2005 utilizando la ponderación que mejores resultados había dado en Abilene [w=sqrt(gravity_model)], con 2.941 matrices de tráfico.

Tabla 2.4. Error entre la TM original y la estimada mediante Tomogravedad

Ponderación	Error MSE
w=constante	-
w=gravity_model	-
W=sqrt(gravity_model)	31,38%

2.3.3. Conclusiones

Es necesario comentar que los autores validan el método Tomogravedad utilizando las matrices de tráfico de la red de AT&T, una red IP como Abilene, pero el caso de GÉANT es distinto, ya que se trata de una red troncal entre redes nacionales que tienen salidas alternativas al resto de Internet.

Aunque los autores declaran en [23] que el método proporciona un ±20% de error (medido como error cuadrático medio, MSE), los resultados de nuestros análisis arrojan unos números por encima de esa cifra; para el caso de escoger como vector de pesos una ponderación proporcional a la raíz del modelo de gravedad, el error se sitúa alrededor del 31% tanto en Abilene como en GÉANT.

La diferencia entre ambos análisis puede ser debida a que a los autores no les fue posible obtener una matriz de tráfico y encaminamiento consistente con las

medidas SNMP de los enlaces, ya que como comentan en [23], no dispusieron de la totalidad de los flujos de tráfico enviados a través de la red debido a problemas con la implementación del colector de flujos de su proveedor, además de que la información de encaminamiento venía en bloques de 24 horas y los flujos de datos eran parciales.

CAPÍTULO 3. SOFTWARE DE MEDIDA Y ANÁLISIS DE FLUJOS DE TRÁFICO

En este capítulo se describe el software utilizado para analizar y procesar las trazas de tráfico obtenidas de RedIRIS. El proceso de instalación de cada aplicación desde el código fuente está recogido en el anexo A.I. Dado que los datos provienen de una red real, se han anonimizado las direcciones IP.

3.1. flow-tools

flow-tools [5, 6] es una colección de programas cuya finalidad es recoger, enviar, procesar y generar informes a partir de los datos procedentes de NetFlow. La información de los flujos en formato flow-tools se recoge y almacena por defecto en orden de byte, por lo que los ficheros son portables entre arquitecturas big y little endian. La diferencia entre estos dos sistemas está en el modo en el que se guardan los datos de más de un byte: big-endian consiste en representar los bytes en el orden "natural": así el valor hexadecimal 0x4A3B2C1D se codificaría en memoria en la secuencia {4A, 3B, 2C, 1D}. En el sistema little-endian el mismo valor se codificaría como {1D, 2C, 3B, 4A}.

Los flujos pueden ser de cuatro tipos:

- Unidireccionales: un flujo por conexión.
- Bidireccionales**:** un flujo contiene tanto datos del origen como la respuesta del destino, por lo que es posible obtener más información que con los unidireccionales, como por ejemplo el *round trip time.*
- De aplicación: clasifican los paquetes por su contenido en base a la cabecera.
- Agregados: conjuntos de flujos agregados según un parámetro concreto, por ejemplo, dirección IP origen – destino.

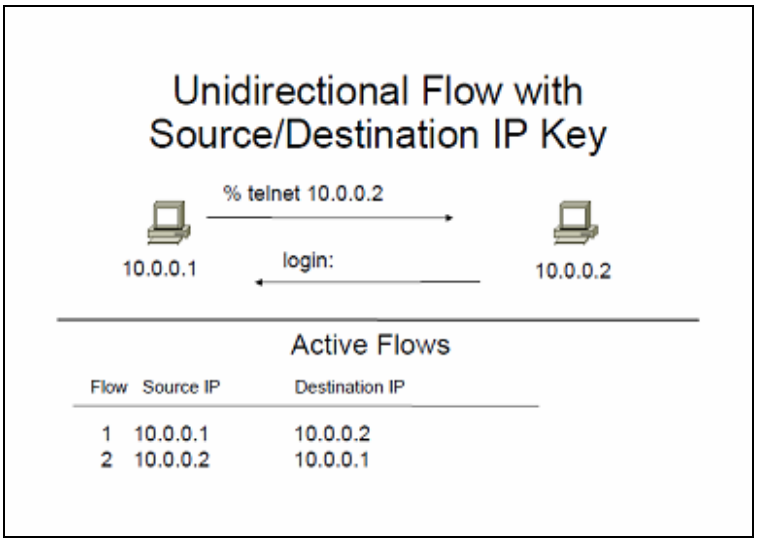


Fig. 3.1 Flujos unidireccionales con el campo dirección IP origen/destino

En este estudio se ha trabajado con la versión 0.68 de las flow-tools donde se cuentan hasta veinticuatro herramientas, de las cuales se han utilizado las siguientes:

- flow-header: muestra meta información de ficheros flow-tools, como por ejemplo la fecha de inicio, el nombre del host que captura los flujos o su versión. (Fig. 3.2)

```
# mode:                 normal
# capture hostname:     hostcollector
# capture start:        Mon Jan 25 12:00:00 2010
# capture end:          Mon Jan 25 12:01:00 2010
# capture period:       60 seconds
# compress:             on
# byte order:           little
# stream version:       3
# export version:       5
# lost flows:           106
# corrupt packets:      0
# sequencer resets:     0
# capture flows:        6892
```

Fig. 3.2 Resultado del comando *flow-header < ft-v05.file*

- flow-merge: une ficheros en formato flow-tools con la característica que mantiene el orden relativo de los flujos antes de combinar los ficheros. Se debe utilizar siempre que se combinan flujos de distintos colectores.
- flow-split: divide un fichero de flow-tools en ficheros más pequeños basándose en el número de flujos o el tiempo transcurrido. Si se desea dividir un único fichero en varios más pequeños, se ha de utilizar conjuntamente con flow-merge, resultando en un comando como el siguiente: *flow-merge -g fichero_ft | flow-split -T300 -o fichero_ft_300s.* La opción *–g* ordena los flujos por tiempo de inicio antes de procesarlos, y con *–T300* se consigue crear ficheros de trescientos segundos.
- flow-export: exporta los datos a los formatos cflowd, pcap, ASCII, MySQL, wire y PGSQL. La figura 3.3 es una exportación parcial de un fichero flow-tools a formato ASCII seleccionando únicamente los campos UNIX_SECS (timestamp, unix epoch), SRCADDR (dirección IP origen), DSTADDR (dirección IP destino), PROT (protocolo). Con la opción *–m* se exportan únicamente los campos deseados.

```
#:unix_secs,srcaddr,dstaddr,prot
1264633200,196.147.120.15,202.88.222.104,6
1264633200,196.147.120.15,72.113.93.106,6
.
1264636656,204.10.157.51,158.213.136.8,6
```

Fig. 3.3 Resultado del comando *flow-export -f2 -mUNIX_SECS,DFLOWS,SRCADDR,DSTADDR,PROT < fichero_ft > flows.ascii.txt*

3.2. nfdump

nfdump [8] (Fig. 3.4) es un conjunto de programas que recogen y procesan datos procedentes de NetFlow desde la línea de comandos, con el objetivo de analizar los datos guardados así como rastrear los flujos para hallar patrones de tráfico.

El funcionamiento del programa se basa en aplicar filtros sobre los datos de NetFlow (Fig. 3.5). A través de ellos se pueden realizar análisis de incidentes, traza de hosts, hallar problemas de red, etc.

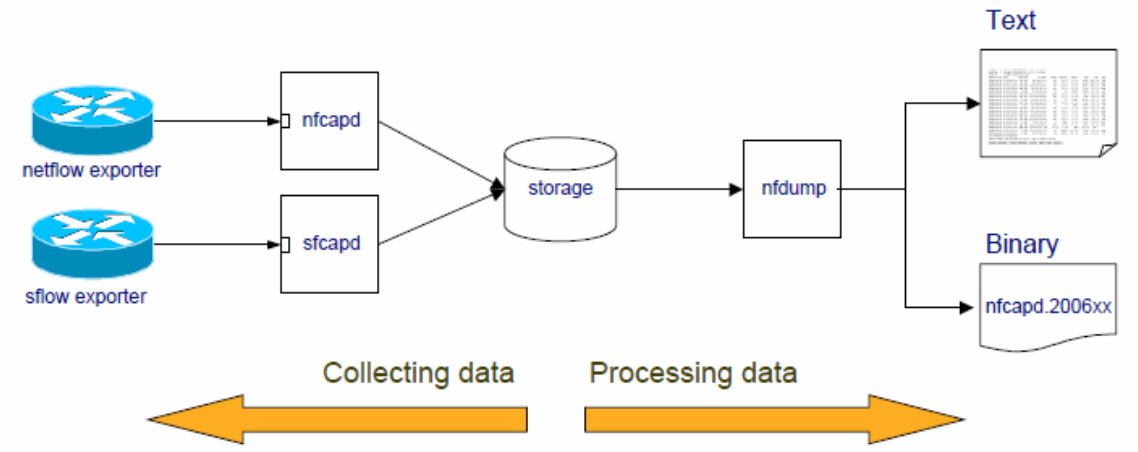


Fig. 3.4 Esquema general de nfdump

Las herramientas que conforman nfdump son las siguientes:

- nfcapd: netflow capture daemon. Lee la información procedente de la red y la guarda en ficheros. Automáticamente rota los ficheros cada *n* minutos (normalmente cada cinco minutos). Es necesario un proceso nfcapd para cada flujo de NetFlow.
- nfdump: netflow dump. Lee la información almacenada en los ficheros creados por nfcapd. Puede mostrar los datos filtrados por diferentes

campos, como por ejemplo puerto origen, destino o dirección IP, así como crear estadísticas a partir de ellos.

- nfprofile: netflow profiler. Lee la información almacenada en los ficheros creados por nfcapd. Filtra los datos de NetFlow de acuerdo a los conjuntos de filtros (*profiles*) y guarda la información filtrada en ficheros para su posterior análisis.
- ft2nfdump: lee y convierte la información de NetFlow que está almacenada en formato flow-tools para que pueda ser procesada por nfdump.

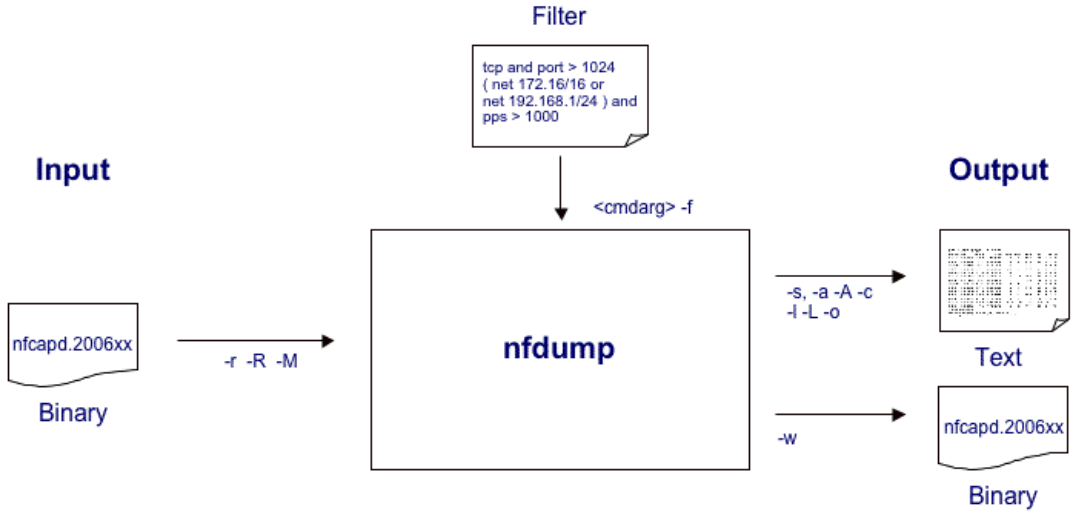


Fig. 3.5 Flujo de trabajo de nfdump

3.2.1. Formatos de salida

nfdump dispone de cuatro formatos de salida fijos: *raw, line, long* y *extended,* aunque es posible especificar la salida deseada utilizando la opción *fmt*. Por defecto, el formato de salida es *line*.

- Raw: el formato *raw* muestra para cada *record* (registro) de NetFlow múltiples líneas, y en cada una de ellas toda la información almacenada en dicho registro (dependiendo del origen, el registro tendrá una información u otra)

```
Flow Record:
  Flags         =              0x00 Unsampled
  size          =                60
  first         =        1264633202 [2010-01-28 00:00:02]
  last          =        1264633204 [2010-01-28 00:00:04]
  msec_first    =               783
  msec_last     =               721
  src addr      =     196.28.182.xxx
```

```
dst addr      =    131.207.30.xxx
src port      =                80
dst port      =             35374
fwd status    =                 0
tcp flags     =              0x10 .A....
proto         =                 6
(src)tos      =                 0
(in)packets   =                 7
(in)bytes     =             10500
input         =               140
output        =               139
src as        =              1273
dst as        =               766
src mask      =                16 196.28.0.0/16
dst mask      =                23 131.207.30.0/23
dst tos       =                 0
direction     =                 0
engine type   =                 0
engine ID     =                 0
```

- Line: muestra un registro de NetFlow por línea. La fecha y la duración del flujo se dan con una resolución de milisegundos. El número de flujos es siempre uno a no ser que se trate de agregados.

```
Date flow start          Duration   Proto   Src IP Addr:Port \
Dst IP Addr:Port         Packets    Bytes   Flows
2010-01-28 00:00:01.052  1.212      TCP     66.56.112.xxx:26815 -> \
194.147.142.xxx:25       22         19701   1
```

- Long: este formato contiene información adicional como flags TCP, Tipo de servicio (ToS), etc.

```
Date flow start          Duration   Proto   Src IP Addr:Port \
Dst IP Addr:Port         Flags      Tos     Packets  Bytes   Flows
2010-01-28 00:00:02.076  0.088      TCP     194.147.142.xxx:6406  -> \
209.51.224.xxx:80        .AP.SF     0       5        966     1
```

- Extended: este formato contiene en cada registro información adicional sobre pps (paquetes por segundo), bps (bits por segundo) y Bpp (bytes por paquete).

```
Date flow start          Duration   Proto   Src IP Addr:Port \
Dst IP Addr:Port         Flags      Tos     Packets  Bytes   pps \
bps     Bpp   Flows
2010-01-28 00:00:02.076  0.088      TCP     194.147.142.xxx:6406  -> \
209.51.224.xxx:80        .AP.SF     0       5        966     56 \
87818   193   1
```

- Formato de salida personalizado: es el más flexible, ya que se puede especificar cómo se quiere la salida. El formato está definido utilizando etiquetas predefinidas:

```
%ts   Start Time - first seen
%te   End Time - last seen
%td   Duration
%pr   Protocol
%sa   Source Address
%da   Destination Address
%sap  Source Address Port
%dap  Destination Address Port
%sp   Source Port
%dp   Destination Port
%nh   Next-hop IP Address
%nhb  BGP Next-hop IP Address
%ra   Router IP Address
%sas  Source AS
%das  Destination AS
%in   Input Interface num
%out  Output Interface num
%pkt  Packets
%byt  Bytes
%fl   Flows
%flg  TCP Flags
%tos  ToS
%bps  bps - bits per second
%pps  pps - packets per second
%bpp  Bpp - Bytes per package
```

En la figura 3.6 se puede observar la salida de un fichero nfdump procesado con las etiquetas %*sa %das %bps %td*

Src IP Addr	Dst AS	bps	Duration
142.81.34.xx	12322	130	14.508
195.142.141.xxx	0	1.3 M	0.004
84.43.0.xxx	13041	45	21.153
190.148.252.xxx	1103	4.4 M	59.774
87.157.102.xxx	65000	171	29.368
163.118.169.xxx	209	24170	0.940

Fig. 3.6 Resultado del comando *nfdump -r nf.v05.file -o "fmt:%sa %das %bps %td"*

3.2.2. Agregación de flujos

Por defecto, los flujos con idéntico protocolo y dirección IP origen y destino así como idénticos puertos origen y destino, son agregados. Sin embargo, este comportamiento se puede cambiar especificando otro tipo de agregación con *–A*. Esta opción acepta cualquier combinación de *srcip, dstip, srcport, dstport*.

Src IP Addr	Dst IP Addr	Bytes
214.135.0.0	194.146.0.0	56829
210.127.0.0	194.109.0.0	311
74.172.0.0	158.46.0.0	31

Fig. 3.7 Agregación de subredes utilizando la opción *-A srcip4/16,dstip4/16 -o "fmt:%sa %da %byt"*

3.3. NfSen

NfSen [8] es un *front-end*, una interfaz web, para nfdump (Fig. 3.8) con el que se puede mostrar información sobre los flujos, paquetes y bytes utilizando RRD (Round Robin Database). NfSen permite procesar datos de NetFlow dentro de un periodo de tiempo definido, crear un histórico de la información analizada o configurar alertas basándose en diferentes condiciones, como por ejemplo ancho de banda utilizado.

NfSen procesa los datos de NetFlow en tiempo real, aunque en este estudio no se ha podido aprovechar esta característica, ya que los conjuntos de datos analizados estaban contenidos en ficheros, no eran *live data*.

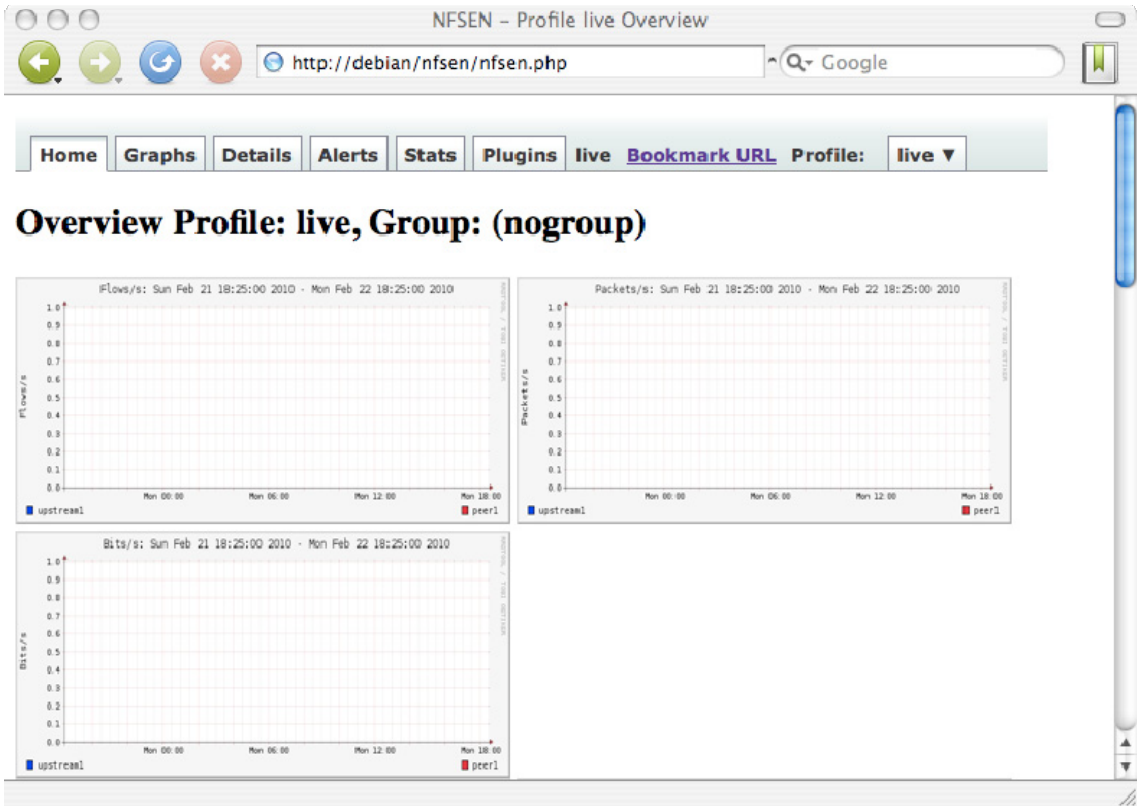


Fig. 3.8 NfSen, front-end web de nfdump

3.4. RRDtool

RRDtool [18] es el acrónimo de *Round Robin Database tool*. Se trata de una aplicación que trabaja con una base de datos que maneja planificación siguiendo un modelo Round Robin, esto es, seleccionando todos los elementos de la base de datos comenzando por el primer elemento de la lista hasta llegar al último y empezando de nuevo desde el primer elemento. Esta técnica trabaja con una cantidad de datos fija, definida en el momento de crear la base de datos, y un puntero al elemento actual.

3.4.1. Funcionamiento

El funcionamiento de una RRD es el siguiente: se trata una base de datos como si fuese un círculo, sobrescribiendo los datos almacenados con anterioridad una vez alcanzada la capacidad máxima de la misma. Esta capacidad máxima dependerá de la cantidad de información que se quiera conservar como historial. Su finalidad principal es el tratamiento de datos temporales y datos seriales como temperaturas, transferencias en redes, cargas del procesador, etc.

Los parámetros que se necesitan definir para monitorizar son variables que primero almacenan los valores y posteriormente los archivan. Puesto que el tiempo es un dato primordial, también se han de crear parámetros específicos para guardar esa información temporal. Debido a su estructura, la definición de una base de datos RRDtool incluye una previsión para tomar acciones específicas en caso de absencia de datos (UNKNOWN). Los *Data Source* (DS), *heartbeat*, *Data Source Type* (DST), *Round Robin Archive* (RRA), y *Consolidation Function* (CF) son algunas de las terminologías relacionadas con las bases de datos RRDtool.

La estructura de una base de datos y la terminología asociada a ella se puede explicar viendo la cabecera de un fichero RRD:

```
filename = "cat-nac.rrd"
rrd_version = "0003"
step = 300
last_update = 1271157208
ds[ds0].type = "COUNTER"
ds[ds0].minimal_heartbeat = 600
ds[ds0].min = 0,0000000000e+00
ds[ds0].max = 1,2500000000e+09
ds[ds0].last_ds = "3086237113054704"
ds[ds0].value = 3,9307405892e+10
ds[ds0].unknown_sec = 0
ds[ds1].type = "COUNTER"
ds[ds1].minimal_heartbeat = 600
ds[ds1].min = 0,0000000000e+00
ds[ds1].max = 1,2500000000e+09
ds[ds1].last_ds = "2704827196804179"
ds[ds1].value = 7,5849125198e+10
ds[ds1].unknown_sec = 0
rra[0].cf = "AVERAGE"
```

```
rra[0].rows = 599
rra[0].cur_row = 3
rra[0].pdp_per_row = 1
rra[0].xff = 5,0000000000e-01
rra[0].cdp_prep[0].value = 1,8467583200e+08
rra[0].cdp_prep[0].unknown_datapoints = 0
rra[0].cdp_prep[1].value = 3,1890215400e+08
rra[0].cdp_prep[1].unknown_datapoints = 0
```

filename: nombre del fichero.

rrdversion: versión de RRD. Puede ser 0001, 0002, 0003 ó 0004. La diferencia entre ellas es el formato con el que se guardan las bases de datos. Hay que hacer notar que las creadas con las versiones 0001-0003 son dependientes de la arquitectura, por lo que no se pueden copiar entre sistemas operativos con distinta ordenación de bytes (big/little endian).

step: el *step* de 300 segundos indica que la base de datos espera nuevos valores cada 300 segundos.

ds (Data Source) es la variable relacionada al parámetro monitorizado en el dispositivo. *ds0* es el nombre con el que se guarda el parámetro en la base de datos. Después de cada intervalo de 300 segundos, un nuevo valor de DS se almacena. Este valor también se llama Primay Data Point (PDP). El tipo de DS está definido por el Data Source Type (DST), pudiendo ser COUNTER, DERIVE, ABSOLUTE o GAUGE. En este caso el DS es del tipo COUNTER, lo que significa que se guardará el cambio del valor durante un periodo de tiempo igual al *step*. DERIVE es lo mismo que COUNTER pero además permite almacenar valores negativos. ABSOLUTE también guarda el cambio del valor, pero asume que el valor previo es cero. La diferencia entre el actual y el valor anterior siempre es igual al valor actual, es decir, almacena el valor actual dividido por el *step.* GAUGE guarda el valor actual.

```
Values       = 300, 600, 900, 1200
Step         = 300 seconds
COUNTER DS   =    1,  1,   1,    1
DERIVE DS    =    1,  1,   1,    1
ABSOLUTE DS  =    1,  2,   3,    4
GAUGE DS     = 300, 600, 900, 1200
```

El *heartbeat* es de 600 segundos. Si la base de datos no obtiene un nuevo PDP en 300 segundos, se esperará otros 300 segundos (en total 600). Si no se recibe ningún PDP en 600 segundos, se guardará el valor UNKNOWN en la base de datos. Los parámetros *ds[ds1].min* y *ds[ds1].max* son los valores mínimos y máximos aceptados. Todos los valores que salgan de ese rango se almacenarán como UNKNOWN.

rra[0] es un archivo Round Robin (Round Robin Archive) en el que se almacena una cantidad fija de CDPs (Consolidated Data Point, en este caso este valor es de 599, *rra[0].rows = 599*). En la RRA también se especifica cuántos PDPs se deben consolidar en un CDP (*rra[0].pdp_per_row = 1*) y qué CF utilizar para ello (*rra[0].cf = "AVERAGE"*). La función de consolidación (CF, Consolidation Function) puede ser AVERAGE, MINIMUM, MAXIMUM y LAST. En este caso

es AVERAGE, por lo que después de que los datos se hayan consolidado, el CDP resultante (valor medio) se almacenará en la RRA. El tiempo total cubierto por la RRA se calcula con la siguiente expresión:

$$\text{Tiempo cubierto} = (\#\text{CDPs guardados}) * (\#\text{PDPs por CDP}) * \text{step} \qquad \textbf{(3.1)}$$

En la `rra[0]` hay definidos 599 CDPs y en cada uno de ellos se guarda un PDP con un *step* de 300 segundos; el tiempo cubierto o periodo de rotación es de 599 * 1 * 300 segundos = 179.700 segundos (2 días, 1 hora y 50 minutos). Transcurrido este tiempo, la siguiente inserción sobrescribirá la entrada más antigua. Para cubrir varios periodos de tiempo y/o utilizar distintas funciones de consolidación, un fichero RRD puede contener múltiples RRAs. Por ejemplo, tal como se puede observar en la figura 3.9, en RedIRIS existen cuatro RRAs, y cada una de ellas almacena un valor medio distinto: 5 minutos, 30 minutos, 2 horas y un día; todas las figuras pertenecen al enlace Nacional – Cataluña y se pueden consultar en el Weathermap de RedIRIS.

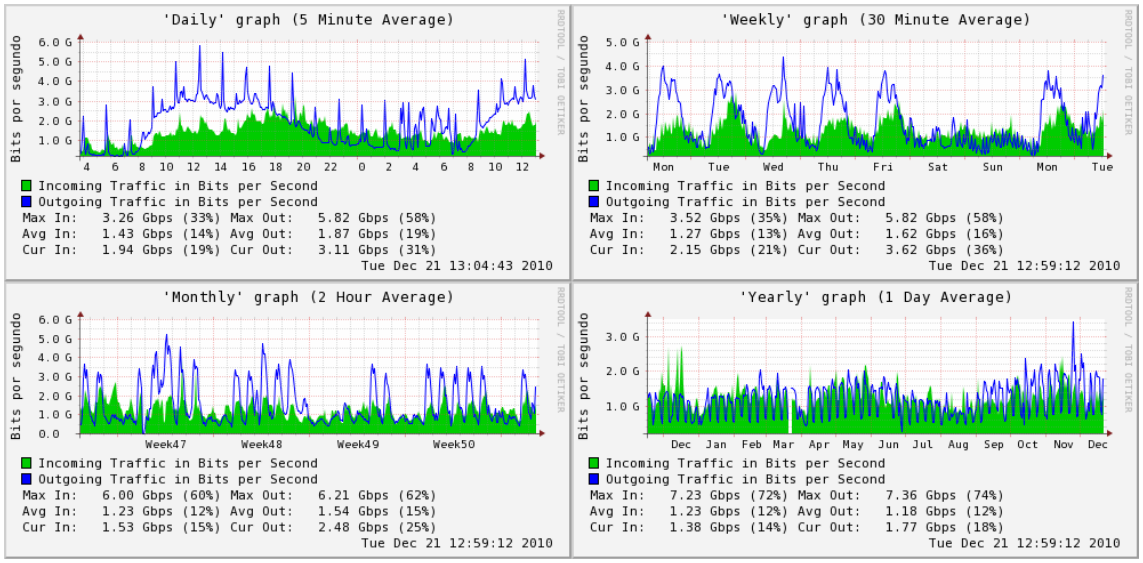


Fig. 3.9 Gráfico de las RRAs del enlace Nacional – Cataluña de RedIRIS

3.4.2. Herramientas

RRDtool está compuesto por catorce herramientas, de las cuales se han utilizado las siguientes:

- rrddump: la función *dump* escribe el contenido de un RRD en un fichero con formato XML (Fig. 3.10) o directamente a stdout.

```
 <rra>
  <cf>AVERAGE</cf>
  <pdp_per_row>1</pdp_per_row>
 <!--
 300 seconds
  -->
 <params>
  <xff>5.0000000000e-01</xff>
  </params>
 <cdp_prep>
 <ds>
  <primary_value>1.8114028088e+08</primary_value>
  <secondary_value>NaN</secondary_value>
  <value>1.8467583200e+08</value>
  <unknown_datapoints>0</unknown_datapoints>
  </ds>
 <ds>
  <primary_value>3.5935852374e+08</primary_value>
  <secondary_value>NaN</secondary_value>
  <value>3.1890215400e+08</value>
  <unknown_datapoints>0</unknown_datapoints>
  </ds>
  </cdp_prep>
<database>
 <!--
 2010-04-11 11:20:00 CEST / 1270977600
  -->
 <row>
  <v>8.0773437249e+07</v>
  <v>6.8187138659e+07</v>
  </row>
.
.
</row>
 <!--
 2010-04-13 13:10:00 CEST / 1271157000
  -->
 <row>
  <v>1.8114028088e+08</v>
  <v>3.5935852374e+08</v>
  </row>
  </database>
  </rra>
```

Fig. 3.10 Extracto de un fichero XML tras utilizar el comando *dump* sobre un RRD

- rrdfetch: la función *fetch* la utiliza internamente *graph* para obtener los datos de los RRDs, pero también se puede utilizar para imprimir por pantalla o redirigir a un fichero la información de la base de datos acotándola temporalmente. En la figura 3.11 se puede observar un extracto de la salida del comando *rrdtool fetch* indicando un *step* de 300 segundos en la RRA AVERAGE para la primera hora del día 1 de enero de 2010.

```
                     ds0                     ds1
1262304000: 8,9237905265e+07 5,9683532502e+07
1262304300: 8,7332470151e+07 8,7767620292e+07
```

Fig. 3.11 Salida del comando *rrdtool fetch fichero.rrd AVERAGE -r 300 -s 20100101 -e start+1h*

- rrdgraph: la función *graph* de RRDtool se utiliza para representar los datos de un RRD.

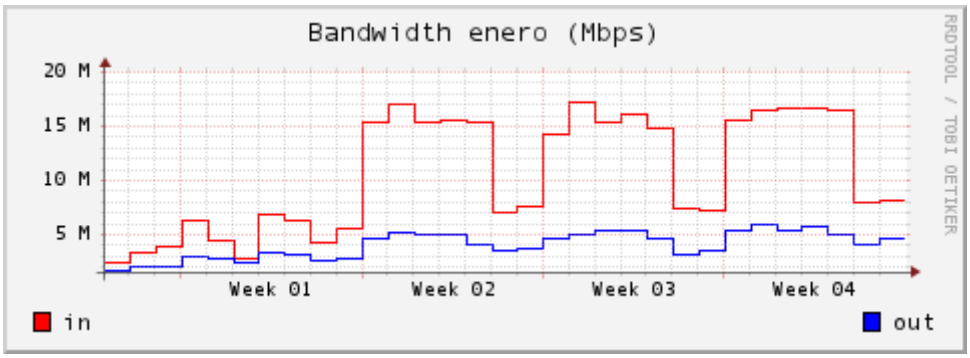


Fig. 3.12 Representación de una RRD mediante *rrdtool graph*

- rrdinfo: la función *info* muestra la información de la cabecera de la base de datos RRD.

```
filename = "cat-nac.rrd"
rrd_version = "0003"
step = 300
last_update = 1271157208
ds[ds0].type = "COUNTER"
ds[ds0].minimal_heartbeat = 600
ds[ds0].min = 0,0000000000e+00
ds[ds0].max = 1,2500000000e+09
ds[ds0].last_ds = "3086237113054704"
ds[ds0].value = 3,9307405892e+10
ds[ds0].unknown_sec = 0
ds[ds1].type = "COUNTER"
ds[ds1].minimal_heartbeat = 600
ds[ds1].min = 0,0000000000e+00
ds[ds1].max = 1,2500000000e+09
ds[ds1].last_ds = "2704827196804179"
ds[ds1].value = 7,5849125198e+10
ds[ds1].unknown_sec = 0
rra[0].cf = "AVERAGE"
rra[0].rows = 599
```

Fig. 3.13 Extracto de la información de la cabecera del fichero *cat-nac.rrd* generada con el comando *rrdtool info*

- rrdxport: similar a *dump*, ya que ambas funciones convierten el contenido de un RRD a XML, pero con *xport* se pueden crear informes, delimitando el tiempo de inicio y final, el *step* o el número máximo de filas (valores). En la figura 3.14 se ha definido el tiempo de exportación mediante las opciones *-s 1262304000 -e 1262307600* de los *data sources* ds0 y ds1 guardados en la RRA AVERAGE.

```
CDEF:aa=in,out,+,8,* XPORT:in:"in bytes" XPORT:aa:"in and out bits"
<?xml version="1.0" encoding="ISO-8859-1"?>
<xport>
  <meta>
    <start>1262390400</start>
    <step>86400</step>
    <end>1262304000</end>
    <rows>0</rows>
    <columns>2</columns>
    <legend>
      <entry>in bytes</entry>
      <entry>in and out bits</entry>
    </legend>
  </meta>
  <data>
  </data>
</xport>
```

Fig. 3.14 Meta data de un RRD tras ejecutar el comando *rrdtool xport -s 1262304000 -e 1262307600 DEF:in=cat-nac.rrd:ds0:AVERAGE DEF:out=cat-nac.rrd:ds1:AVERAGE*

CAPÍTULO 4. ANÁLISIS DE LOS DATOS DE REDIRIS

Desde RedIRIS nos facilitaron las bases de datos de los contadores SNMP, durante una semana, junto con los datos globales de NetFlow (esto es, con la información enviada por todos los routers de la red) durante un mes. Al tratarse de datos sensibles, todas las direcciones IP se han anonimizado. Para poder calcular las matrices de tráfico es necesario también conocer la topología de la red. El objetivo de este capítulo es analizar, procesar y crear, a partir de los datos obtenidos vía NetFlow, ficheros de una duración de cinco minutos conteniendo el volumen de tráfico intercambiado entre pares de nodos origen - destino.

4.1. RedIRIS

RedIRIS [15] es una red de comunicaciones de alta capacidad, que permite a la comunidad académica y científica española participar en proyectos de investigación nacionales e internacionales donde se requieran servicios avanzados de comunicaciones. Su red troncal, denominada RedIRIS-10 (Fig. 4.1), comunica las instituciones universitarias y de investigación españolas entre sí, a través de las redes académicas autonómicas.

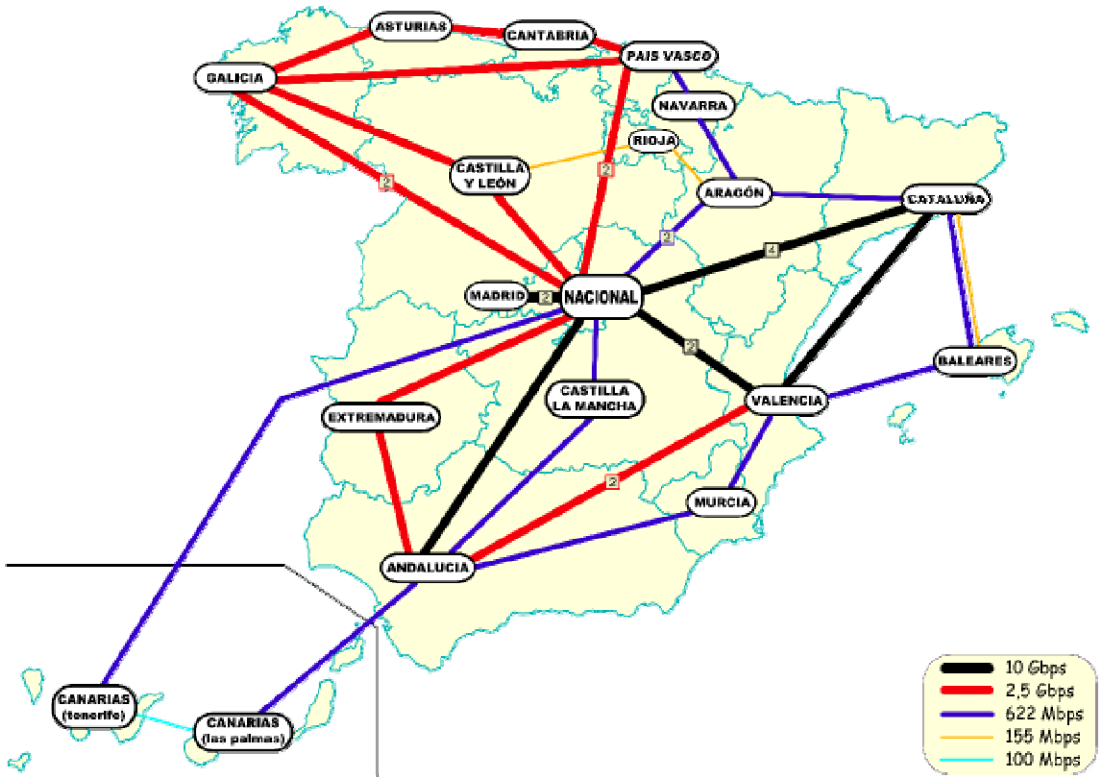


Fig. 4.1 Topología de RedIRIS-10

RedIRIS-10 proporciona acceso a la red de investigación mundial a través de la red pan-europea GÉANT2, una infraestructura de red de fibra oscura con un punto de presencia por país, que interconecta a 33 redes nacionales de investigación. GÉANT2 es una red híbrida donde se soportan servicios de conmutación de circuitos y conmutación de paquetes. Proporciona además el acceso a las redes de investigación de otras zonas del mundo como Internet2 (USA), Canarie (Canada), RedCLARA (América Latina), EUMEDCONNECT (Norte de África), UbuntuNet (Este y Sur de África), TENET (Sur de África), TEIN2 (Asia Pacífica), SINET (Japón), CERNET, CSTNET (en China) o ERNET (India).

La conectividad externa de RedIRIS se complementa con el acceso a la Internet Comercial Global a través de dos salidas a dos proveedores de ámbito internacional, Telia y Global Crossing. Además, RedIRIS tiene presencia en el punto de intercambio[1] nacional ESPANIX ubicado en Madrid y en el ubicado en Barcelona, CATNIX.

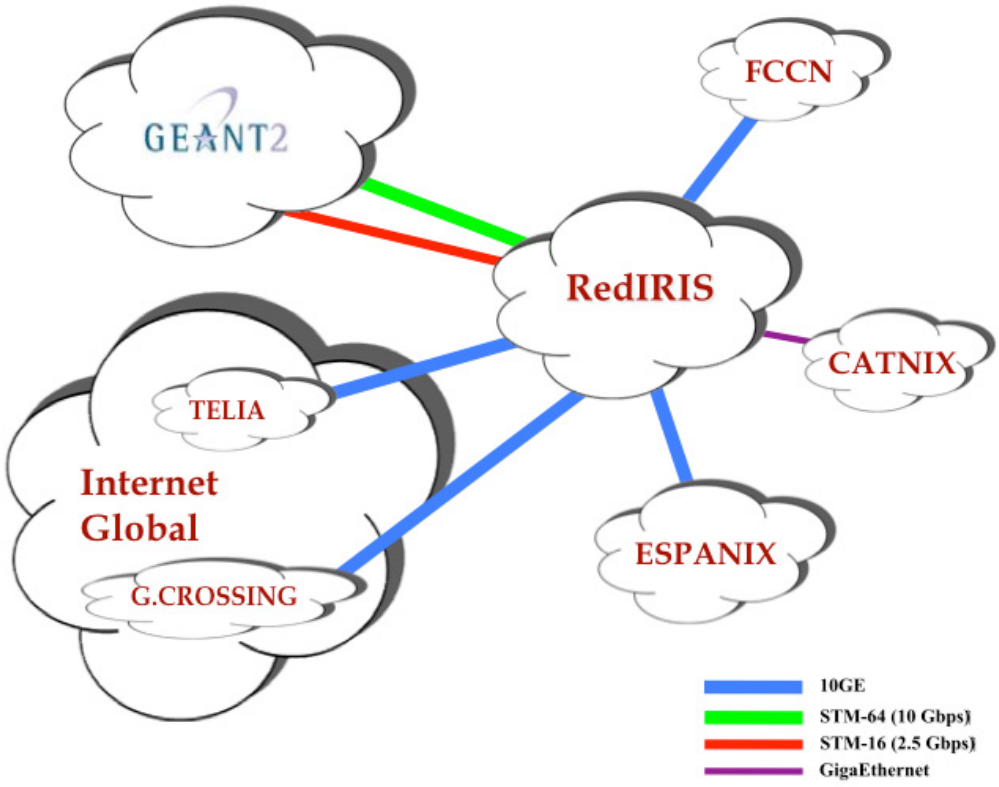


Fig. 4.2 Conectividad externa de RedIRIS.

[1] Un punto de intercambio (en inglés IX o IXP, *Internet Xchange Point*) es una infraestructura de telecomunicaciones donde los ISP intercambian el tráfico de Internet entre sus sistemas autónomos

4.1.1. Nodos y encaminamiento

La topología actual de RedIRIS-10 es mallada y cuenta con Puntos de Presencia en todas las comunidades autónomas.

Equipo Origen	Interface Origen	IP Origen	Equipo Destino	Interface Destino	IP Destino
EB-Badajoz0	so-1/1/0	130.206.x.x	EB-Sevilla0	so-1/1/0	130.206. x.x
EB-Badajoz0	so-1/0/0	130.206. x.x	EB-IRIS4	so-6/1/3	130.206. x.x
EB-Barcelona0	ge-0/0/0	130.206. x.x	EB-Valencia0	ge-0/0/0	130.206. x.x
EB-Barcelona0	ge-1/0/0	130.206. x.x	EB-IRIS2	ge-6/0/0	130.206. x.x
EB-Barcelona0	so-2/0/0	130.206. x.x	EB-Zaragoza0	so-1/1/0	130.206. x.x
EB-Barcelona0	so-3/0/0	130.206. x.x	EB-Palma0	so-0/0/0	130.206. x.x
EB-Barcelona0	so-2/0/1	130.206. x.x	EB-Palma0	so-1/3/0	130.206. x.x
EB-Bilbao0	so-5/0/0	130.206. x.x	EB-Santander0	so-2/0/0	130.206. x.x
EB-Bilbao0	so-0/0/0	130.206. x.x	EB-Santiago0	so-2/0/0	130.206. x.x
EB-Bilbao0	so-1/0/0	130.206. x.x	EB-Pamplona0	so-0/0/0	130.206. x.x
EB-Bilbao0	as0	130.206. x.x	EB-IRIS4	as0	130.206. x.x
EB-Bilbao0	so-3/0/0		EB-IRIS4	so-1/0/0	
EB-Bilbao0	so-7/0/0		EB-IRIS4	so-6/1/2	
EB-CiudadReal0	so-0/0/0	130.206. x.x	EB-Sevilla0	so-1/0/0.0	130.206. x.x
EB-CiudadReal0	so-1/3/0	130.206. x.x	EB-IRIS4	so-1/1/1	130.206. x.x
EB-Madrid0	ge-1/0/0		Nodo Nacional		
EB-Madrid0	ge-1/1/0		Nodo Nacional		
EB-Oviedo0	so-3/0/0	130.206. x.x	EB-Santander0	so-3/0/0	130.206. x.x
EB-Oviedo0	so-2/0/0	130.206. x.x	EB-Santiago0	so-1/0/0	130.206. x.x
EB-Pamplona0	so-1/3/0	130.206. x.x	EB-Zaragoza0	so-0/0/0	130.206. x.x
EB-Santiago0	so-0/0/0	130.206. x.x	EB-Valladolid0	so-0/0/0	130.206. x.x
EB-Santiago0	as0	130.206. x.x	EB-IRIS2	as0	130.206. x.x
EB-Santiago0	so-5/0/0		EB-IRIS2	so-6/1/0	
EB-Santiago0	so-6/0/0		EB-IRIS2	so-6/1/3	
EB-Sevilla0	ge-0/0/0	130.206. x.x	EB-IRIS2	ge-7/0/0	130.206. x.x
EB-Sevilla0	as0	130.206. x.x	EB-Valencia0	as0	130.206. x.x
EB-Sevilla0	so-1/2/0		EB-Valencia0	so-2/3/0	130.206. x.x
EB-Sevilla0	so-1/3/0		EB-Valencia0	so-2/0/0	
EB-Sevilla0	so-1/0/1	130.206. x.x	EB-Murcia0	so-0/0/0	130.206. x.x
EB-Sevilla0	so-1/0/2	130.206. x.x	Eb-LasPalmas0	so-1/0/0	130.206. x.x
EB-Tenerife0	ge-0/0/0.37	130.206. x.x	Eb-LasPalmas0	ge0/0/0.37	130.206. x.x
EB-Tenerife0	so-1/0/0	130.206. x.x	EB-IRIS4	so-1/1/3	130.206. x.x
EB-Valencia0	ge-0/1/0	130.206. x.x	EB-IRIS4	ge-6/0/0	130.206. x.x
EB-Valencia0	so-2/1/0	130.206. x.x	EB-Murcia0	so-0/1/0	130.206. x.x
EB-Valencia0	so-2/1/1	130.206. x.x	EB-Palma0	so-0/3/0	130.206. x.x
EB-Valladolid0	so-4/0/0	130.206. x.x	EB-Rioja0	so-0/0/0	130.206. x.x
EB-Valladolid0	so-0/1/0	130.206. x.x	EB-IRIS4	so-6/1/1	130.206. x.x
EB-Zaragoza0	so-1/2/0	130.206. x.x	EB-Rioja0	so-0/0/1	130.206. x.x
EB-Zaragoza0	as0	130.206. x.x	EB-IRIS4	as1	130.206. x.x
EB-Zaragoza0	so-1/3/0		EB-IRIS4	so-1/1/2	
EB-Zaragoza0	so-1/0/0		EB-IRIS4	so-1/1/0	

Fig. 4.3 Enlaces troncales de RedIRIS

En la figura 4.4 se encuentra la tabla de encaminamiento de uno de los nodos de la red.

```
10.0.0.4+65535+1+255.0.0.0+14+10.0.0.4
127.0.0.1+65535+1+255.255.255.255+21+127.0.0.1
128.0.0.4+65535+1+192.0.0.0+14+128.0.0.4
130.206.x.x.+65535+1+255.255.255.248+125+130.206.x.x
130.206.x.x +65535+1+255.255.255.255+16+130.206.x.x
130.206.x.x +65535+1+255.255.255.224+119+130.206.x.x
130.206.x.x +65535+1+255.255.255.252+126+130.206.x.x
130.206.x.x +65535+1+255.255.255.252+117+130.206.x.x
```

Fig. 4.4 Tabla de encaminamiento anonimizada de un nodo

4.1.2. Descripción de los conjuntos de datos

En este estudio se analizan y procesan las trazas *ft* (NetFlow) globales facilitadas por RedIRIS correspondientes al mes de enero de 2010, utilizando las herramientas *flow-tools* y *nfdump.* El objetivo es generar, para cada cinco minutos, el volumen de tráfico/paquetes/flujos intercambiados entre dos prefijos y de esta manera poder calcular las matrices de tráfico.

Desde RedIRIS también nos enviaron los datos SNMP de los enlaces troncales, muy útiles para comparar los modelos de predicción de las TMs con las medidas reales de los enlaces. Los datos SNMP están almacenados en formato RRD (Round Robin Database), por lo que también se ha tenido que trabajar con la herramienta *RRDtool* para analizar, procesar y representar la información relativa a los contadores SNMP.

Se ha de indicar que los datos proporcionados por RedIRIS no están anonimizados, tal y como sí lo estaban las matrices de tráfico de GÉANT, pero en este caso ha habido un problema con el desfase temporal entre la información aportada por NetFlow (enero 2010) y los datos de los enlaces SNMP, ya que aunque en las bases de datos RRD hay registros desde febrero del año 2008, la granuralidad de los contadores SNMP para el mes de enero de 2010 es de dos horas, insuficiente para comparar las matrices de tráfico estimadas mediante NetFlow, ya que éstas tendrán un periodo de cinco minutos.

Como se verá en el punto 4.1.3, las trazas de NetFlow tienen una duración de 3456 segundos (57 minutos y 36 segundos), haciendo un total de 25 trazas por día. El periodo de almacenamiento de los contadores SNMP varía entre 300 segundos y un día.

4.1.3. Procesado de las trazas

Antes de procesar las trazas *ft* de NetFlow es necesario conocer la información de exportador, versión de flujo y la hora de inicio y final. Para conseguir estos datos se ha utilizado la herramienta *flow-header.* Al ser datos globales todas las trazas contienen la misma información, por lo que accediendo a la meta información de cabecera de cualquier fichero se obtienen los mismos datos.

En la figura 4.5 está la cabecera de la traza perteneciente a las 00:00 horas del jueves 28 de enero de 2010: el colector es una máquina llamada *fraguelrock*, y se exportan los flujos en versión 5 con un periodo de captura de 3456 segundos.

```
# mode:                 normal
# capture hostname:     fraguelrock
# capture start:        Thu Jan 28 00:00:00 2010
# capture end:          Thu Jan 28 00:57:36 2010
# capture period:       3456 seconds
# compress:             on
# byte order:           little
# stream version:       3
# export version:       5
# lost flows:           3108
# corrupt packets:      0
# sequencer resets:     0
# capture flows:        9164193
```

Fig. 4.5 Comando *flow-header* < *ft-v05.2010-01-28.000000+0100*

La versión 5 de los flujos contiene los campos clave especificados en la figura 1.6, y entre ellos se encuentran dirección IP origen, dirección IP destino y bytes enviados, que son los requeridos para generar las matrices de tráfico. Aunque existen herramientas propias de las *flow-tools* para obtener estos datos (por ejemplo, *flow-filter* o *flow-nfilter*), se decidió convertir las trazas al formato nfdump para poder analizarlas y procesarlas mediante la aplicación de filtros específicos y conseguir una tabla formada únicamente por dirección IP origen, destino y volumen de tráfico.

Antes de convertir las trazas *ft* de 3456 segundos (57 minutos y 36 segundos) al formato nfdump, se ha escrito un script en *bash* para dividirlas en ficheros de 300 segundos (5 minutos) utilizando conjuntamente las aplicaciones *flow-merge* y *flow-split.* El comando para realizar la conversión es el siguiente:

```
flow-merge -g ft-v05.2010-01-28.000000+0100 | flow-split -T300 -o
5min.2010-01-28_00_00_00-00_05_00
```

Con la opción -g de *flow-merge* se ordenan los flujos contenidos en el fichero `ft-v05.2010-01-28.000000+0100` por tiempo de inicio antes de procesarlos, y se concatena con *flow-split* especificando la opción -T300 para crear los ficheros `5min.2010-01-28_00_00_00-00_05_00`, los cuales tendrán una duración de 300 segundos.

En el script quedaría de la siguiente manera:

```
#!/bin/bash
FILES="*"
for f in $FILES
do
echo "Dividiendo traza $f"
      flow-merge -g $f | flow-split -T300 –o 5min.$f
done
```

El siguiente paso es exportar los ficheros de 300 segundos al formato nfdump mediante la herramienta *ft2nfdump.* Hay que hacer notar que para el mes de

enero de 2010 se generan 8.928 ficheros *ft* de 5 minutos, por lo que se ve necesario escribir de nuevo un script que automatice el proceso de conversión:

```
#!/bin/bash
FILES="*"
for f in $FILES
do
echo "Convirtiendo fichero ft $f"
      ft2nfdump -r $f | nfdump -w nf.$f
done
```

Con la opción -r de *ft2nfdump* se le pasa el fichero *ft* a convertir y con la opción -w de *nfdump* se escribe el fichero en formato nfdump.

Una vez creado el fichero ya se puede procesar con *nfdump* aplicando los filtros para obtener en la salida la información relativa a dirección IP origen, destino y bytes enviados (Fig. 4.6). Idealmente hubiera sido muy útil conocer los campos Next-hop IP (router al que se enviará el paquete) y BGP next-hop IP (por dónde sale el paquete), pero en las trazas estos datos siempre aparecen como 0.0.0.0 (Fig. 4.7)

Src IP Addr	Dst IP Addr	Bytes
193.144.x.x	209.85.x.x	52
193.144.x.x	79.110.x.x	40
193.144.x.x	79.110.x.x	844

Fig. 4.6 Primeros registros de la salida del comando *nfdump -r nf.5min.2010-01-28_00_00_00-00_05_00.0 -o "fmt:%sa %da %byt"*

Src IP Addr	Dst IP Addr	Next-hopIP	BGPnext-hopIP	Bytes
193.144.x.x	209.85.x.x	0.0.0.0	0.0.0.0	52
193.144.x.x	79.110.x.x	0.0.0.0	0.0.0.0	40
193.144.x.x	79.110.x.x	0.0.0.0	0.0.0.0	844

Fig. 4.7 Primeros registros de la salida del comando *nfdump -r nf.5min.2010-01-28_00_00_00-00_05_00.0 -o "fmt:%sa %da %nh %nhb %byt"*

Durante los cinco primeros minutos del 28 de enero de 2010 se capturaron 856.367 flujos, por lo que para calcular las matrices de tráfico es necesario generar agregación de flujos por prefijo; para ello se pueden utilizar 8, 16 ó 24 bits. En la figura 4.8 se ha realizado una agregación de 16 bits.

Src IP Addr	Dst IP Addr	Bytes
216.x.0.0	193.x.0.0	70237
212.x.0.0	193.x.0.0	302
213.x.0.0	64.x.0.0	168

Fig. 4.8 Primeros registros de la salida del comando *nfdump -r nf.5min.2010-01-28_00_00_00-00_05_00.0 -A srcip4/16,dstip4/16 -o "fmt:%sa %da %byt"*

Para comparar las matrices de tráfico con las medidas reales de los enlaces se utilizan los datos de los contadores SNMP, almacenados en formato RRD. En este estudio se ha trabajado con la aplicación *RRDtool*, que permite analizar, procesar y representar bases de datos Round Robin.

El método de análisis de las RRD es para todos los ficheros idéntico; como ejemplo se han seleccionado los enlaces Cataluña-Nacional (cat-nac.rrd) y Madrid-Nacional (mad-nac1.rrd) por ser dos de los que transportan mayor tráfico.

El primer paso es acceder a la información de la RRD mediante la herramienta *info.* En el punto A.III.1. del anexo se puede ver la información de la cabecera del fichero cat-nac.rrd; de ahí se extraen los siguientes datos de relevancia:

- Nombre del fichero: cat-nac.rrd
- Versión de rrd: 0003
- Step: 300 segundos
- El tipo de DS está definido por el Data Source Type (DST), siendo éste del tipo COUNTER, lo que significa que se guardará el cambio del valor durante un periodo de tiempo igual al *step*, que en este caso es de 300 segundos con un *heartbeat* de 600.
- Los *data sources* ds0 y ds1 son las interfaces entrada y salida, respectivamente.
- La base de datos se compone de cuatro RRA, Round Robin Archive. Para hallar el tiempo cubierto por cada uno de ellos se utiliza la fórmula (3.1).
    - RRA de 300 segundos: está formada por 599 CDPs (Consolidated Data Point), y en cada CDP se consolida un PDP (Primary Data Point), creando un ciclo de 179.700 segundos, 2.995 minutos ó 2 días, 1 hora y 50 minutos.
    - RRA de 1.800 segundos, 30 minutos: está formada por 700 CDPs, y en cada CDP se consolidan 6 PDPs, creando un ciclo de 1.260.000 segundos, 21.000 minutos, 350 horas ó 14 días, 13 horas y 30 minutos.

- RRA de 7.200 segundos, 2 horas: está formada por 775 CDPs, y en cada CDP se consolidan 24 PDPs, creando un ciclo de 5.580.000 segundos, 93.000 minutos, 1.550 horas ó 64 días y 13 horas.

- RRA de 86.400 segundos, 1 día: está formada por 796 CDPs, y en cada CDP se consolidan 288 PDPs, creando un ciclo de 68.774.400 segundos, 1.146.240 minutos, 19.104 horas ó 796 días.

Una vez se conoce el ciclo Round Robin de los RRA, hay que encontrar la fecha exacta de inicio y finalización de recogida de datos para cada uno de ellos. Esta información se consigue convirtiendo la RRD en formato XML mediante el comando *rrdtool dump* (Fig. 4.9).

```
<rra>
  <cf>AVERAGE</cf>
  <pdp_per_row>1</pdp_per_row>
 <!--
 300 seconds
  -->
<params>
  <xff>5.0000000000e-01</xff>
  </params>
<cdp_prep>
<ds>
  <primary_value>1.8114028088e+08</primary_value>
  <secondary_value>NaN</secondary_value>
  <value>1.8467583200e+08</value>
  <unknown_datapoints>0</unknown_datapoints>
  </ds>
<ds>
  <primary_value>3.5935852374e+08</primary_value>
  <secondary_value>NaN</secondary_value>
  <value>3.1890215400e+08</value>
  <unknown_datapoints>0</unknown_datapoints>
  </ds>
  </cdp_prep>
<database>
 <!--
 2010-04-11 11:20:00 CEST / 1270977600
  -->
<row>
  <v>8.0773437249e+07</v>
  <v>6.8187138659e+07</v>
  </row>
.
.
</row>
 <!--
 2010-04-13 13:10:00 CEST / 1271157000
  -->
<row>
  <v>1.8114028088e+08</v>
  <v>3.5935852374e+08</v>
  </row>
  </database>
  </rra>
```

Fig. 4.9 Extracto de la RRA de 300 segundos de cat-nac.rrd, utilizando el comando *rrdtool dump cat-nac.rrd > cat-nac.xml && cat cat-nac.xml*

En el punto A.III.2. del anexo se encuentra la RRD del fichero cat-nac.rdd en formato XML con las fechas de inicio y final de cada RRA; de ahí se extrae la siguiente información sobre la actualización de la base de datos:

- RRA 300 segundos (5 min)
  - Fecha inicio: 2010-04-11 11:20:00 CEST / 1270977600
  - Fecha finalización: 2010-04-13 13:10:00 CEST / 1271157000

- RRA 1.800 segundos (30 minutos)
  - Fecha inicio: 2010-03-29 23:30:00 CEST / 1269898200
  - Fecha finalización: 2010-04-13 13:00:00 CEST / 1271156400

- RRA 7.200 segundos (2 horas)
  - Fecha inicio: 2010-02-07 23:00:00 CET / 1265580000
  - Fecha finalización: 2010-04-13 12:00:00 CEST / 1271152800

- RRA 86.400 segundos (1 día)
  - Fecha inicio: 2008-02-08 01:00:00 CET / 1202428800
  - Fecha finalización: 2010-04-13 02:00:00 CEST / 1271116800

Es posible representar la RRD gráficamente mediante la herramienta *graph* de RRDtool; en las figuras 4.10, 4.11 y 4.12 se puede observar la granuralidad de cada RRA dependiendo de la fecha seleccionada junto con el comando para crear el gráfico.

Línea de comandos para la figura 4.10:

```
 rrdtool graph graph_cat-nac.rrd_20080101_x_grid.png --start 20080101 --
end 20100630 --title "cat-nac.rrd" --height=200 --width=1000 --
vertical-label="MBps" DEF:in=cat-nac.rrd:ds0:AVERAGE
LINE:in#ff0000:in\\n DEF:out=cat-nac.rrd:ds1:AVERAGE
LINE:out#0000ff:out\\l
```

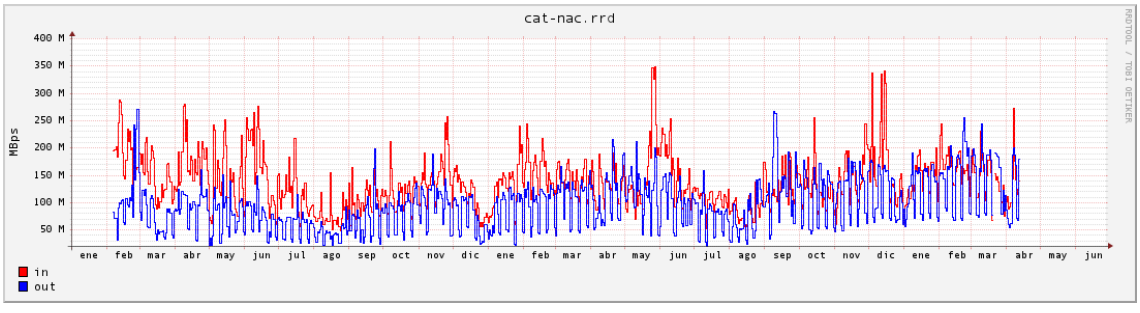


Fig. 4.10 Enero de 2008 a junio de 2010. Sólo aparecen los datos comprendidos entre la fecha de inicio y finalización de la RRA de 1 día.

Línea de comandos para la figura 4.11:

```
rrdtool graph graph_cat-nac.rrd_300s.png --start 1270977600 --end
1271157000 --title "11/04/2010 09:20:00 GMT - 13/04/2010 11:10:00 GMT
cat-nac.rrd" --height=200 --width=1000 --vertical-label="MBps"
DEF:in=cat-nac.rrd:ds0:AVERAGE LINE:in#ff0000:in\\n DEF:out=cat-
nac.rrd:ds1:AVERAGE LINE:out#0000ff:out\\l
```

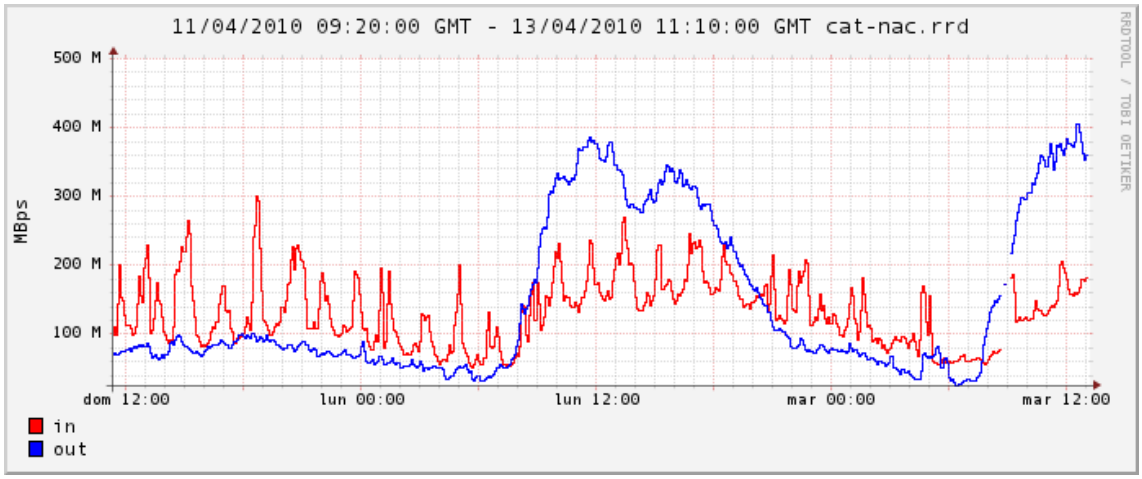


Fig. 4.11 RRA de 300 segundos de cat-nac.rrd, de 11/04/2010 a 13/04/2010.

Línea de comandos para la figura 4.12:

```
rrdtool graph graph_mad-nac1.rrd_marzo_1a_semana_grid%X_2.png --start
20100301 --end start+7d -x HOUR:12:DAY:1:DAY:1:86400:%X --title
"20100301-20100307 mad-nac1.rrd" --height=200 --width=700 --vertical-
label="MBps" DEF:in=mad-nac1.rrd:ds0:AVERAGE LINE:in#ff0000:in\\n
DEF:out=mad-nac1.rrd:ds1:AVERAGE LINE:out#0000ff:out\\l
```

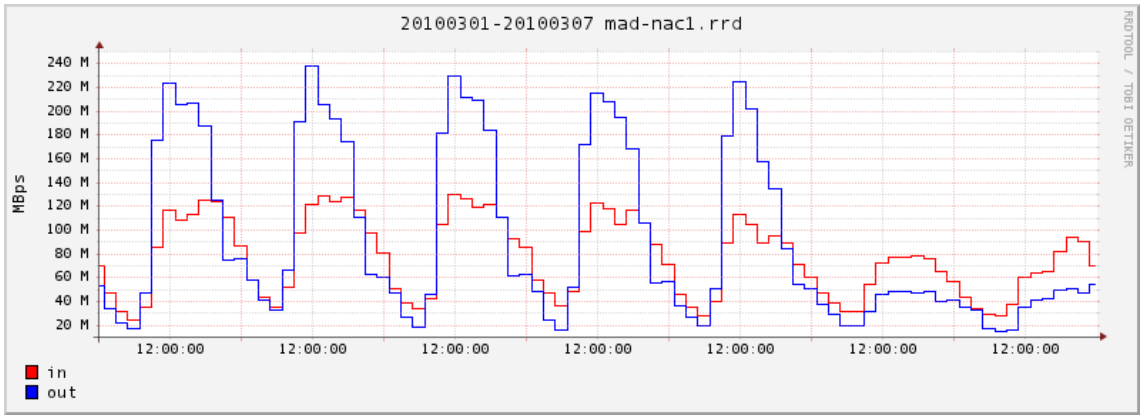


Fig. 4.12 RRA de 2 horas de mad-nac1.rrd, de 01/03/2010 a 07/03/2010

CAPÍTULO 5. CONCLUSIONES Y LÍNEAS FUTURAS

5.1. Conclusiones

Los objetivos de este trabajo eran dos: aplicar el método de Tomogravedad a las matrices de tráfico de Abilene y GÉANT y tratar los datos proporcionados por RedIRIS para llegar a generar la matriz de tráfico de su red. Se ha presentado el método de Tomogravedad, el formato y la arquitectura de los registros de NetFlow y de las bases de datos Round Robin.

Se ha comprobado la buena aproximación que supone la generación de matrices de tráfico para las redes Abilene y GÉANT mediante el método de Tomogravedad. Este estudio, con los datos concretos de ambas redes, es la primera vez que se realizaba y ha verificado que el error producido por el método de inferencia es superior al afirmado por sus autores en [23]. La diferencia entre ambos análisis podría ser debida a que a los autores no les fue posible obtener una matriz de tráfico y encaminamiento consistente con las medidas SNMP de los enlaces.

Se ha realizado una introducción a las diferentes aplicaciones utilizadas para crear matrices de tráfico a partir de los registros de NetFlow proporcionados por RedIRIS, así como de la información de la carga de los enlaces, con la finalidad de comparar las matrices estimadas con los datos de los contadores SNMP. Igualmente, se han documentado los pasos de instalación de todas las aplicaciones desde el código fuente junto con los formatos de salida de cada herramienta.

El análisis de los datos proporcionados por RedIRIS, tanto de los registros de NetFlow como de las bases de datos RRD, comenzaba desde cero. En referencia a NetFlow, se ha conseguido dividir los flujos en periodos de cinco minutos y convertirlos al formato nfdump con el objetivo de aplicar los filtros necesarios para agregar flujos por prefijo, obteniendo así la relación entre dirección IP origen, IP destino y bytes enviados, datos necesarios para crear una matriz de tráfico.

Mediante RRDtool se ha realizado un análisis de las bases de datos RRD de RedIRIS, teniendo como finalidad el comparar las matrices generadas a partir de los datos de NetFlow con la información almacenada en los contadores SNMP. También se han representado gráficamente los archivos RRD como aparecen en el *Weathermap* de RedIRIS [15].

Se ha creado la documentación necesaria referente a la parte de procesado mecánico y transformación de trazas, lo que permitirá que próximos TFCs se centren exclusivamente en el análisis de datos.

5.2. Líneas futuras

Los resultados aportados en este trabajo, incluyendo la instalación desde el código fuente de las aplicaciones, podrían ser el punto de partida de futuros estudios que tengan como objetivo la modelización de la matriz de demanda en redes IP. Así mismo, con la experiencia adquirida en este TFC se podría pasar a hacer una campaña de medidas sistemática y de inferencia de matrices de tráfico de RedIRIS, así como de su predicción mediante modelos matemáticos, tal y como ya se está realizando [3].

También se podría seguir la línea iniciada en los TFCs de Josep Xavier Torres [20] e Isaac Balasch [2] sobre el análisis multirresolución de matrices de tráfico.

En referencia al impacto medioambiental, un buen modelo de matrices de demanda permite más eficiencia en muchos sentidos, incluido el energético; se podrían conseguir ahorros de energía al evitar congestiones, minimizar el impacto de un fallo de red o incluso optimización de rutas que permitieran desconectar equipos.

BIBLIOGRAFÍA

[1] Abilene Network http://www.internet2.edu/network/

[2] Balasch, I., "Anàlisi Espai-Temporal Multiresolució de Matrius de Teletrànsit". Treball Fi de Carrera EPSC (2010).

[3] Escriche, S., "Predicción de tráfico de redes de comunicaciones" Treball Fi de Carrera EPSC (2010).

[4] Eum, S., Murphy, J., Harris, R. J., "TomoKruithof vs Tomogravity for Backbone Networks". *RMIT University, Melbourne, Victoria 3000, Australia* (2004).

[5] Flow-tools http://www.splintered.net/sw/flow-tools/

[6] Flow-tools Tutorial, SANOG 6
http://www.sanog.org/resources/sanog6/gaurab-sanog6-flow-tools.pdf

[7] GÉANT Network http://www.geant.net/

[8] Haag, B.P., "nfdump & NfSen". *SWINOG-12* (2006).

[9] Introduction to Cisco IOS NetFlow - A Technical Overview
http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_white_paper0900aecd80406232.html

[10] MATLAB - The Language Of Technical Computing
htttp://www.mathworks.com/products/matlab

[11] Medina, A., Fraleigh, C., Taft, N., Bhattacharyya, S., Diot, C.. "A taxonomy of IP traffic matrices". *SPIE ITCOM: Scalability and Traffic Control in IP Networks II* (Agosto 2002).

[12] Medina, A., Taft, N., Salamatian, K., Bhattacharyya, S., Diot, C., "Traffic Matrix Estimation: Existing Techniques and New Directions". *SIGCOMM'02* (2002).

[13] NetFlow, Wikipedia http://en.wikipedia.org/wiki/Netflow

[14] Nucci, A., Sridharan, A., Taft, N., "The problem of synthetically generating IP traffic matrices: Initial recommendations". *SIGCOMM Computer Communication Review. 35 (3)* (2005).

[15] RedIRIS http://www.rediris.es/lared/

[16] Rincón, D., Roughan, M., and Willinger, W., "Towards a Meaningful MRA of Traffic Matrices". *Internet Measurement conference. Proceedings of the 8th ACM SIGCOMM conference on Internet Measurements*, 331-336 (2008).

[17] Roughan, M. Thorup, M. and Zhang, Y., “Traffic Engineering with Estimated Traffic Matrices”. *Proceedings of the USENIX/ACM Internet Measurement Conference,* 248-258 (2003).

[18] RRDtool Home Site http://oss.oetiker.ch/rrdtool/

[19] RRDtool, Wikipedia http://en.wikipedia.org/wiki/RRDtool

[20] Torres Haba, J. X., “Modelling of traffic matrices with multiresolution analysis techniques”. Treball Fi de Carrera EPSC (2009).

[21] TOTEM Project http://totem.run.montefiore.ulg.ac.be

[22] Uhlig, S., Quoitin, B., Lepropre, J., Balon, S., “Providing public intradomain traffic matrices to the research community”, *ACM SIGCOMM Computer Communication review*. 36 (1). pp 83-86. (2006).

[23] Zhang, Y., Roughan, M., Duffield, N., Greenberg, A, “Fast Accurate Computation of Large Scale IP Traffic Matrices from Link Loads”. *ACM SIGMETRICS Performance Evaluation Review*, 206-217 (2003).

GLOSARIO

AS	Autonomous System
BGP	Border Gateway Protocol
GM	Gravity Model
IP	Internet Protocol
ISP	Internet Service Provider
IXP	Internet Exchange Point
LAN	Local Area Network
MAN	Metropolitan Area Network
MIB	Management Information Bases
MSE	Mean Square Error
NREN	National Research and Education Network
PoP	Point of Presence
QoS	Quality of Service
RRA	Round Robin Archive
RRD	Round Robin Database
SCTP	Stream Control Transmission Protocol
SNMP	Simple Network Management Protocol
SVD	Singular Value Descomposition
TCP	Transmission Control Protocol
TM	Traffic Matrix
TOTEM	Toolbox for Traffic Engineering Methods
ToS	Type of Service
UDP	User Datagram Protocol
WAN	Wide Area Network
XML	Extensible Markup Language

ANEXOS

A.I. Instalación del software desde el código fuente

En este punto se detalla el proceso de instalación desde el código fuente de *flow-tools*, *RRDtool*, *nfdump* y *NfSen*.

A.I.1. flow-tools

Para compilar *ft2nfdump*, aplicación perteneciente a *nfdump* que lee, convierte y guarda los datos procedentes de *NetFlow*, se necesita el código fuente de la versión 0.68 de las *flow-tools* ya que la cabecera necesaria para ello no está en la 0.68.4.3; es por esto que se han de obtener las dos versiones de los siguientes enlaces:
http://www.splintered.net/sw/flow-tools/ (v0.68)
http://code.google.com/p/flow-tools/downloads/detail?name=flow-tools-0.68.4.3.tar.bz2&can=2&q= (v0.68.4.3)

Se descomprimen ambos ficheros

```
ivan@debian:~$ tar -zxvf flow-tools-0.68.tar.gz
ivan@debian:~$ tar -jxvf flow-tools-0.68.4.3.tar.bz2
```

Cambiar al directorio donde se encuentra el código fuente de la versión 0.68.4.3

```
ivan@debian:~$ cd flow-tools-0.68.4.3/
```

Configurar, compilar e instalar las flow-tools

```
ivan@debian:~/flow-tools-0.68.4.3$ ./configure
ivan@debian:~/flow-tools-0.68.4.3$ make
ivan@debian:~/flow-tools-0.68.4.3$ su
Password:
debian:/home/ivan/flow-tools-0.68.4.3# make install
```

Una vez instaladas, hay que copiar el fichero libft.a al directorio flow-tools-0.68.4.3/lib

```
ivan@debian:~$ cp /usr/local/flow-tools/lib/libft.a flow-tools-
0.68.4.3/lib/
```

También es necesario copiar el fichero ftbuild.h del código fuente de flow-tools-0.68/src a flow-tools-0.68.4.3/src, y así poder compilar posteriormente *nfdump* incluyendo la aplicación *ft2nfdump*

```
ivan@debian:~$ cp flow-tools-0.68/src/ftbuild.h flow-tools-
0.68.4.3/src/
```

A.I.2. RRDtool

Para la instalación de *RRDtool* se necesitan una serie de paquetes que se instalan automáticamente como dependencias gracias al programa *aptitude*, front-end del *Advanced Packaging Tool* (APT) de Debian:

```
debian:~# aptitude install libpango1.0-dev libxml2-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading extended state information
Initializing package states... Done
Reading task descriptions... Done
The following NEW packages will be installed:
  debhelper{a} html2text{a} libcairo2-dev{a} libdirectfb-dev{a} libdirectfb-
  extra{a} libexpat1-dev{a} libfontconfig1-dev{a} libfreetype6-dev{a}
  libglib2.0-dev{a} libice-dev{a} libjpeg62-dev{a} libmpeg3-1{a} libmpeg3-
  dev{a} libpango1.0-dev libpixman-1-dev{a} libpng12-dev{a} libpthread-
  stubs0{a} libpthread-stubs0-dev{a} libsm-dev{a} libsysfs-dev{a} libx11
  dev{a} libxau-dev{a} libxcb-render-util0-dev{a} libxcb-render0-dev{a}
  libxcb1-dev{a} libxdmcp-dev{a} libxext-dev{a} libxft-dev{a} libxml2-dev
  libxrender-dev{a} x11proto-core-dev{a} x11proto-input-dev{a} x11proto-kb
  dev{a} x11proto-render-dev{a} x11proto-xext-dev{a} xtrans-dev{a}
```

El código fuente de *RRDtool* está disponible en http://oss.oetiker.ch/rrdtool/pub/?M=D y se necesita la versión 1.4.2 (rrdtool-1.4.2.tar.gz)

Se descomprime el fichero y se cambia al nuevo directorio para configurar *RRDtool*

```
ivan@debian:~$ tar -zxvf rrdtool-1.4.2.tar.gz
ivan@debian:~$ cd rrdtool-1.4.2/
```

Se compilará el programa en /home/ivan/rrdtool-1.4.2 (donde está el código fuente) y se instalará en /usr/local/rrdtool, directorio que es necesario crear:

```
ivan@debian:~/rrdtool-1.4.2$ su
Password:
debian:/home/ivan/rrdtool-1.4.2# mkdir /usr/local/rrdtool
```

Una vez se tiene la estructura correcta para la compilación e instalación de *RRDtool*, se han de guardar las dos variables de entorno:

```
ivan@debian:~/rrdtool-1.4.2$ BUILD_DIR=/home/ivan/rrdtool-1.4.2/
ivan@debian:~/rrdtool-1.4.2$ INSTALL_DIR=/usr/local/rrdtool/
```

Configurar, compilar e instalar

```
ivan@debian:~/rrdtool-1.4.2$ ./configure --prefix=$INSTALL_DIR
ivan@debian:~/rrdtool-1.4.2$ make
ivan@debian:~/rrdtool-1.4.2$ su
Password:
debian:/home/ivan/rrdtool-1.4.2# make install
```

A.I.3. nfdump

Para la instalación de *nfdump* son necesarios los siguientes paquetes con sus respectivas dependencias: *flex, byacc y libghc6-zlib-dev*

```
debian:~# aptitude install flex byacc libghc6-zlib-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading extended state information
Initializing package states... Done
Reading task descriptions... Done
The following NEW packages will be installed:
  byacc flex ghc6{a} haskell-utils{a} libghc6-zlib-dev libgmp3-dev{a}
  libgmpxx4ldbl{a} libreadline5-dev{a}
```

El código fuente de *nfdump* se encuentra disponible en http://sourceforge.net/projects/nfdump/
La versión compatible con las *flow-tools* es la 1.6 (nfdump-1.6.tar.gz)

Se descomprime el fichero y se cambia al nuevo directorio para configurar nfdump

```
ivan@debian:~$ tar -zxvf nfdump-1.6.tar.gz
ivan@debian:~$ cd nfdump-1.6/
```

Al configurar *nfdump* con la opción nfprofile, necesaria para instalar *NfSen*, `configure` espera encontrar *RRDtool* en /usr con rrd.h en /usr/include y librrd en /usr/lib
El directorio donde está *RRDtool* se puede especificar con la opción `--enable-nfprofile --with-rrdpath=[/usr]`, pero tanto la cabecera rrd.h como las librerías hay que copiarlas a los directorios correspondientes

```
debian:~# cp /usr/local/rrdtool/include/rrd.h /usr/include/
debian:~# cp /usr/local/rrdtool/lib/librrd* /usr/lib/
```

Configurar *nfdump* incluyendo el convertidor *ft2nfdump* y el nfprofile necesario para *NfSen*:

```
ivan@debian:~/nfdump-1.6$ ./configure --enable-nfprofile --with-
rrdpath=/usr/local/rrdtool/lib/ --enable-ftconv --with-
ftpath=/home/ivan/flow-tools-0.68.4.3/
```

Compilar e instalar

```
ivan@debian:~/nfdump-1.6$ make
ivan@debian:~/nfdump-1.6$ su
Password:
debian:/home/ivan/nfdump-1.6# make install
```

A.I.4. NfSen

Para la instalación de *NfSen* son necesarios los siguientes paquetes con sus respectivas dependencias: *php5 php-net-socket libapache2-mod-php5 libapache2-mod-perl2 librrds-perl*

```
debian:~# aptitude install php5 php-net-socket libapache2-mod-php5
libapache2-mod-perl2 librrds-perl
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading extended state information
Initializing package states... Done
Reading task descriptions... Done
The following NEW packages will be installed:
  apache2{a} apache2-mpm-prefork{a} apache2-utils{a} apache2.2-bin{a}
  apache2.2-common{a} libapache2-mod-perl2 libapache2-mod-php5
  libapache2-reload-perl{a} libapr1{a} libaprutil1{a}
  libaprutil1-dbd-sqlite3{a} libaprutil1-ldap{a} libbsd-resource-perl{a}
  libdb4.8{a} libdevel-symdump-perl{a} libperl5.10{a} librrds-perl php-net-
  socket php-pear{a} php5 php5-cli{a} php5-common{a} php5-suhosin{a}
```

El código fuente de NfSen se encuentra disponible en http://sourceforge.net/projects/nfsen/
La versión compatible con *nfdump* es la 1.3.2 (nfsen-1.3.2.tar.gz)

Se descomprime el fichero y se cambia al nuevo directorio

```
ivan@debian:~$ tar -zxvf nfsen-1.3.2.tar.gz
ivan@debian:~$ cd nfsen-1.3.2/
```

Hacer una copia del fichero de configuración nfsen-dist.conf que hay dentro de etc renombrándolo a nfsen.conf

```
ivan@debian:~/nfsen-1.3.2$ cp etc/nfsen-dist.conf etc/nfsen.conf
```

Editar nfsen.conf de acuerdo a las necesidades del sistema

```
ivan@debian:~/nfsen-1.3.2$ vi etc/nfsen.conf
```

Modificar $BASEDIR para que apunte a una partición con suficiente espacio, ya que es donde se almacenarán los datos procedentes de *NetFlow*

```
# Required for default layout
$BASEDIR = "/data1/nfsen";
```

Definir con qué usuario se ejecutará nfcapd y los procesos *NetFlow*

```
# BASEDIR unrelated vars:
#
# Run nfcapd as this user
# This may be a different or the same uid than your web server.
# Note: This user must be in group $WWWGROUP, otherwise nfcapd
#       is not able to write data files!
$USER    = "netflow";

# user and group of the web server process
# All netflow processing will be done with this user
$WWWUSER  = "www-data";
$WWWGROUP = "www-data";
```

Al tener instalado el servidor Apache, tanto el usuario como el grupo www-data ya existen en el sistema, por lo que sólo es necesario crear el usuario netflow y añadirlo al grupo www-data

```
debian:~# useradd -g www-data netflow
```

Para comprobar que el usuario pertenece al grupo

```
debian:~# groups netflow
netflow : www-data
```

Ejecutar como root el script de instalación pasándole como opción el fichero de configuración

```
debian:/home/ivan/nfsen-1.3.2# ./install.pl etc/nfsen.conf
Check for required Perl modules: All modules found.
Upgrade from version '1.3.2' installed at Mon Feb 22 17:37:53
2010
Setup NfSen:
Version: 1.3.2: $Id: install.pl 24 2007-11-21 09:12:03Z phaag $
```

Indicar la ruta donde se encuentra perl, por defecto /usr/bin/perl

```
Perl to use: [/usr/bin/perl]
```

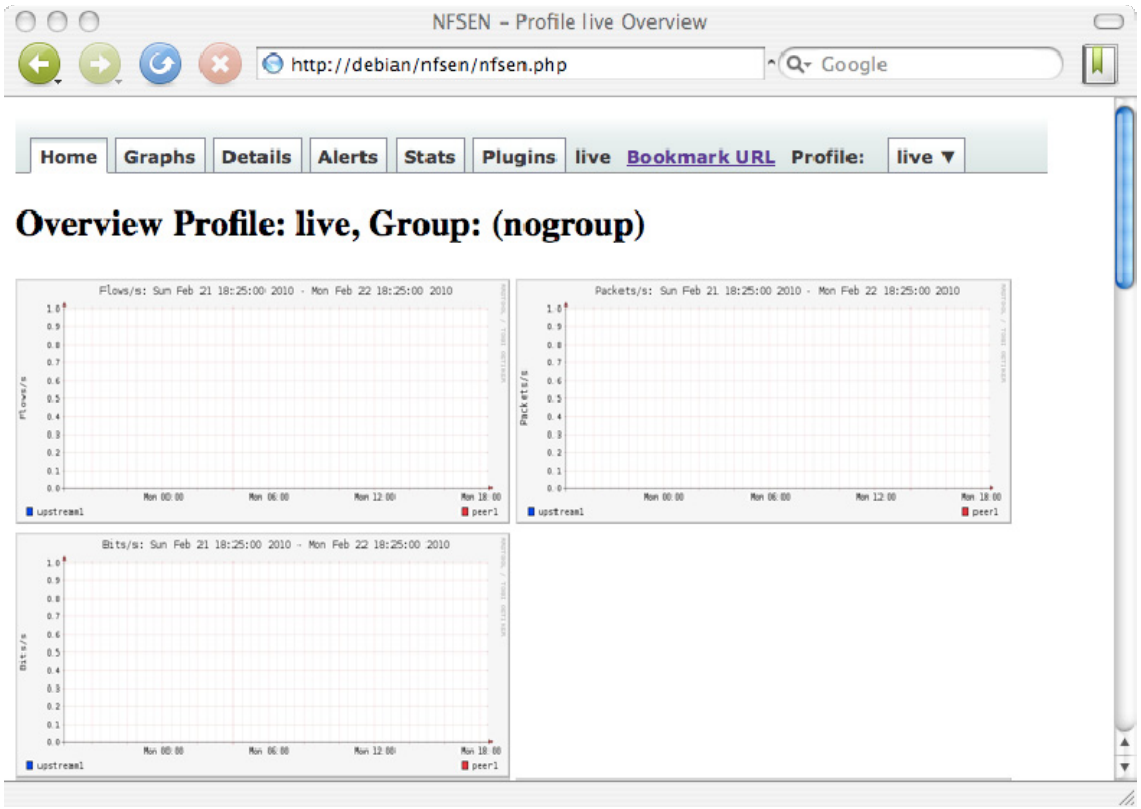
Una vez que se ha instalado correctamente, enlazar $BASEDIR/bin/nfsen a /etc/init.d/nfsen

```
debian:~# ln -s /data1/nfsen/bin/nfsen /etc/init.d/nfsen
```

Iniciar *NfSen*

```
debian:~# /etc/init.d/nfsen start
Starting nfcapd: upstream1[2445] peer1[2448].
Starting nfsend.
debian:~#
```

Abrir en un navegador la dirección http://nombreservidor/nfsen/nfsen.php



Si no se tenía instalado con anterioridad Apache con soporte para PHP, es recomendable reiniciar el servicio y recargar los módulos de PHP. La forma más sencilla es reiniciando la máquina.

Posible error de *NfSen*:

Si en el fichero nfsen.conf se ha especificado otro usuario o grupo para ejecutar nfcapd y los procesos *NetFlow*, es posible que no se tenga permisos de lectura y escritura en el directorio donde NfSen guarda los datos. Para solucionarlo se ha de modificar el fichero $BASEDIR/libexec/Nfcomm.pm y cambiar los permisos del socket para que cualquier usuario pueda leer y escribir.

```
debian:~# vi /data1/nfsen/libexec/Nfcomm.pm
:%s/0660/0666/g [enter]
:wq![enter]
```

Reiniciar *NfSen*

```
debian:~# /etc/init.d/nfsen reload
```

A.II. Aplicación del método de Tomogravedad

A.II.1. Abilene

A.II.1.1. Análisis de una TM con ponderación constante

Para aplicar el código *wlse* (weighted least-squares estimate of the TM) de M. Roughan es necesario haber calculado previamente new_routing_matrix (A), link_loads (x), la solución del modelo de gravedad (tg) y escoger un vector de pesos (w)

wlse_ROUGHAN.m

```
% weighted least-squares estimate of the TM
% Input:
% A matrix A in constraints x=A*t
% x vector x in constraints x=A*t
% tg initial gravity model solution
% w weight vector
% Output:
% t estimated traffic matrix (as a vector)
% that minimizes |(t-tg)./w|
% among all t's that minimize |A*t-x|
function [t] = wlse(A,x,tg,w)
% equivalently transform x=A*t into
% xw=Aw*tw, where tw=(t-tg)./w
xw = x - A*tg;
[r, c] = size(A);
Aw = A .* repmat(w', r, 1);
% solve tw=Aw*tw by computing the pseudo-
% inverse of matrix Aw (through svd)
tw = pinv(full(Aw)) * xw;
% transform tw back to t
t = tg + w .* tw;
```

Calcular new_routing_matrix (A) y link_loads (x)

```
>> load abileneWeightsInfAutolinks
>> load abileneAdjacencyAutoLinks
>> [routing_matrix, sparseroutingmatrix, pathlengths, pathcorrelationmatrix,
pathsparsecorrelationmatrix]=routingMatrix2(abileneWeightsInfAutolinks,abileneAdjacencyAutoli
nks);

%Buscar los enlaces
link_origins=[];
link_destinations=[];
index=1:size(abileneAdjacencyAutolinks,1);
for i=index;
        for j=index;
                if(abileneAdjacencyAutolinks (i,j))
                        link_origins=[link_origins i];
                        link_destinations=[link_destinations j];
```

```
                end;
        end;
end;

%Reducir la matriz a los nodos que están presentes
new_routing_matrix=[];
index=1:length(link_origins);
for i=index;
        row_index=(link_origins(i)-1)*12+ link_destinations(i);% row index
        row_contents=routing_matrix(row_index,:);
        new_routing_matrix=[new_routing_matrix; row_contents];
end;


%Reshape:permite convertir una matriz en otra (por ejemplo, una 12x12 en 144x1)
tm1_abilene_vector = reshape(tm1_abilene', 1, 144);

link_loads = new_routing_matrix *tm1_abilene_vector';
```

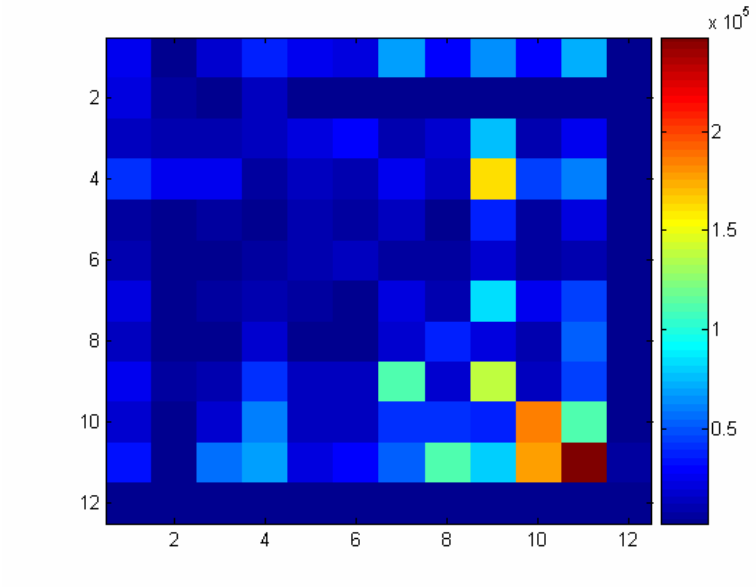
Una vez se conocen new_routing_matrix (A) y link_loads (x), el siguiente paso es aplicar el modelo de gravedad a la matriz original.

Cargar la TM mediante la función *abileneTMparser*

```
>>    tm1_abilene=abileneTMparser('C:\Users\ivan\tfc\matlab\abilene-TM\TM\2004\09\TM-2004-09-06-1645.xml')
```

Representar la TM

```
>> imagesc(tm1_abilene);colorbar;
```

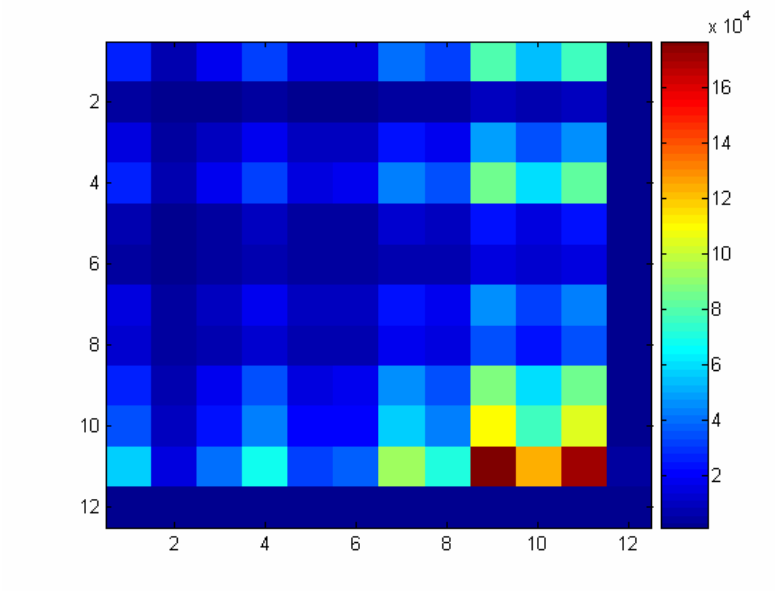


Aplicar Modelo de Gravedad a la TM

```
>> gmtm1_abilene=GravityModel(tm1_abilene);
```

Representar la TM

```
>> imagesc(gmtm1_abilene);colorbar
```



Después de haber calculado new_routing_matrix (A), link_loads (x) y la solución del modelo de gravedad de la matriz original (gmtm1_abilene), es necesario convertir tg mediante la función *reshape*, ya que no se puede multiplicar A*tg debido a que el número de columnas de A no coincide con el número de filas de tg.

```
>> A=new_routing_matrix;
>> x=link_loads;
>> w=ones(144,1); %vector de pesos constante
>> tg=gmtm1_abilene;
>> tg1_abilene_vector = reshape(tg, 1, 144);
>> tg=tg1_abilene_vector'
```

En este punto ya se aplica la función *wlse* para conseguir la matriz estimada (tm')

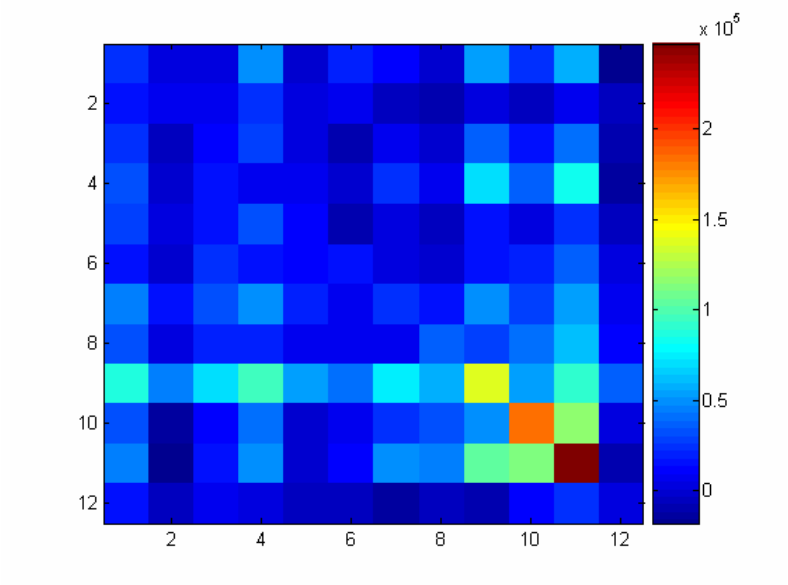
```
>> estimated_traffic_matrix_as_a_vector=wlse_ROUGHAN(A,x,tg,w)
```

Volver a convertir tg en una matriz 12x12

```
>> estimated_traffic_matrix=reshape(estimated_traffic_matrix_as_a_vector,12,12)
```

Representar la TM estimada

```
>> imagesc(estimated_traffic_matrix);colorbar;
```

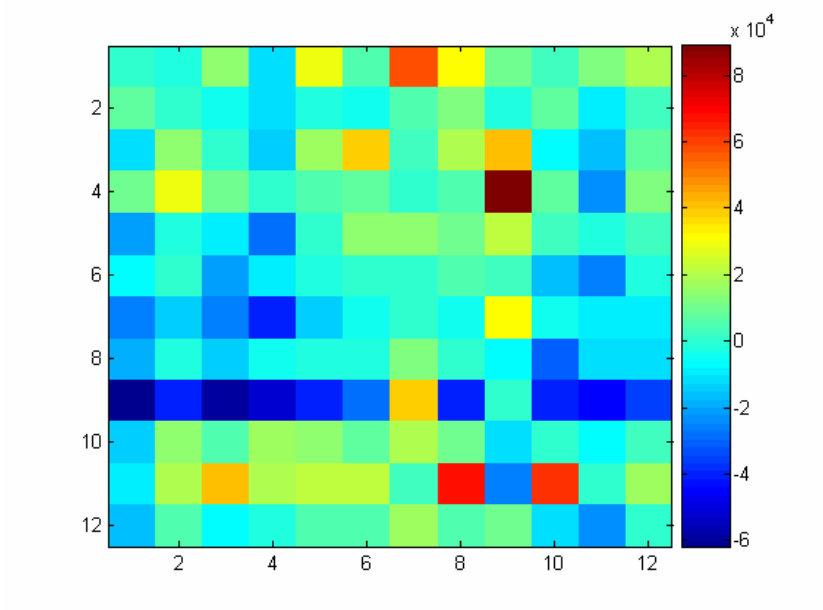


Calcular el error

```
>> error=tm1_abilene-estimated_traffic_matrix;
```

Representar el error

```
>> imagesc(error);colorbar
```




```
>> mse=sum(sum(error.^2))
>> original=sum(sum(tm1_abilene^2))
>> mse/original

ans =

    0.0438
```

La matriz estimada difiere únicamente en un 4,38% (medido como error MSE) de la potencia de la matriz original.

A.II.1.2. Análisis del mes 05 de 2004 con ponderación constante

```
format short g    %aumentar precisión
path='C:\Users\ivan\tfc\matlab\abilene-TM\TM\2004\05\TM-2004-05'
link_loads_matrix=[];
tm_vector_matrix=[];
new_routing_matrix=[];

load abileneWeightsInfAutolinks
load abileneAdjacencyAutoLinks
[routing_matrix, sparseroutingmatrix, pathlengths, pathcorrelationmatrix,
pathsparsecorrelationmatrix]=routingMatrix2(abileneWeightsInfAutolinks,abileneAdjacencyAutoli
nks);

%Buscar los enlaces
link_origins=[];
link_destinations=[];
index=1:size(abileneAdjacencyAutolinks,1);
for i=index;
        for j=index;
                if(abileneAdjacencyAutolinks (i,j))
                        link_origins=[link_origins i];
                        link_destinations=[link_destinations j];
end;
        end;
end;

%Reducir la matriz a los nodos que están presentes
new_routing_matrix=[];
index=1:length(link_origins);
for i=index;
        row_index=(link_origins(i)-1)*12+ link_destinations(i);% row index
row_contents=routing_matrix(row_index,:);
new_routing_matrix=[new_routing_matrix; row_contents];
end;

tm_matrix_matrices=[];
estimated_traffic_matrix_matrices=[];
gmtm_abilene_matrix=[];
estimated_traffic_matrix_matrices_error=[];

for day=1:31;
   daystring=num2str(day,'%02d');
   for hour=0:23;
      hourstring=num2str(hour,'%02d');
      for minutes=0:5:55;
         minutestring=num2str(minutes,'%02d');
         tmpath=strcat(path,'-',daystring,'-',hourstring,minutestring,'.xml');
         tm=AbileneTMparser(tmpath);
          %Reshape:permite convertir una matriz en otra (por ejemplo, una 12x12 en 144x1)
         tm_vector = reshape(tm', 1, 144);
         link_loads = new_routing_matrix *tm_vector';

%aplicar modelo de gravedad
```

```
gmtm_abilene=GravityModel(tm);
gmtm_abilene_matrix=[gmtm_abilene_matrix gmtm_abilene];

%link_loads_matrix=[link_loads_matrix link_loads]; %es útil guardar link_loads_matrix?
%tm_vector_matrix=[tm_vector_matrix; tm_vector]; %es útil guardar tm_vector_matrix?
tm_matrix_matrices=[tm_matrix_matrices tm];

%roughan
A=new_routing_matrix;
x=link_loads;
w=ones(144,1);
tg=gmtm_abilene;
%No se puede multplicar A*tg porque el nº de columnas de A no coincide con el nº de filas de
tg. Hay que convertir tg.
tg_vector = reshape(tg, 1, 144);
tg=tg_vector';

estimated_traffic_matrix_as_a_vector=wlse_ROUGHAN(A,x,tg,w);
estimated_traffic_matrix=reshape(estimated_traffic_matrix_as_a_vector,12,12);

%calcular error entre matriz original (tm) y estimada
error=tm-estimated_traffic_matrix;
mse=sum(sum(error.^2));
original=sum(sum(tm^2));
estimated_traffic_matrix_error=mse/original;

%guardar el error de cada una de las matrices estimadas
estimated_traffic_matrix_matrices_error=[estimated_traffic_matrix_matrices_error
estimated_traffic_matrix_error];

%guardar todas las matrices estimadas
estimated_traffic_matrix_matrices=[estimated_traffic_matrix_matrices estimated_traffic_matrix];
end;
   end;
   fprintf(strcat('Day: ',daystring,'\n'));
end;

%Calcular error medio de todas las matrices estimadas
>> mean(estimated_traffic_matrix_matrices_error)

ans =

    0.32562
```

El error MSE entre las TMs originales y las estimadas es del 32,56% utilizando como vector de pesos w=constante.

A.II.1.3. Análisis del mes 05 de 2004 con ponderación w=gravity_model

```
format short g   %aumentar precisión
path='C:\Users\ivan\tfc\matlab\abilene-TM\TM\2004\05\TM-2004-05'
link_loads_matrix=[];
tm_vector_matrix=[];
new_routing_matrix=[];

load abileneWeightsInfAutolinks
load abileneAdjacencyAutoLinks
```

```
[routing_matrix, sparseroutingmatrix, pathlengths, pathcorrelationmatrix,
pathsparsecorrelationmatrix]=routingMatrix2(abileneWeightsInfAutolinks,abileneAdjacencyAutoli
nks);

%Buscar los enlaces
link_origins=[];
link_destinations=[];
index=1:size(abileneAdjacencyAutolinks,1);
for i=index;
        for j=index;
                if(abileneAdjacencyAutolinks (i,j))
                        link_origins=[link_origins i];
                        link_destinations=[link_destinations j];
                end;
        end;
end;

%Reducir la matriz a los nodos que están presentes
new_routing_matrix=[];
index=1:length(link_origins);
for i=index;
        row_index=(link_origins(i)-1)*12+ link_destinations(i);% row index
        row_contents=routing_matrix(row_index,:);
        new_routing_matrix=[new_routing_matrix; row_contents];
end;

tm_matrix_matrices=[];
estimated_traffic_matrix_matrices=[];
gmtm_abilene_matrix=[];
estimated_traffic_matrix_matrices_error=[];

for day=1:31;
   daystring=num2str(day,'%02d');
   for hour=0:23;
      hourstring=num2str(hour,'%02d');
      for minutes=0:5:55;
         minutestring=num2str(minutes,'%02d');
         tmpath=strcat(path,'-',daystring,'-',hourstring,minutestring,'.xml');
         tm=AbileneTMparser(tmpath);
          %Reshape:permite convertir una matriz en otra (por ejemplo, una 12x12 en 144x1)
         tm_vector = reshape(tm', 1, 144);
         link_loads = new_routing_matrix *tm_vector';

          %aplicar modelo de gravedad
          gmtm_abilene=GravityModel(tm);
          gmtm_abilene_matrix=[gmtm_abilene_matrix gmtm_abilene];

          %link_loads_matrix=[link_loads_matrix link_loads]; %es útil guardar link_loads_matrix?
          %tm_vector_matrix=[tm_vector_matrix; tm_vector]; %es útil guardar tm_vector_matrix?
          tm_matrix_matrices=[tm_matrix_matrices tm];

%wlse
A=new_routing_matrix;
x=link_loads;
gmtm_abilene_vector=reshape(gmtm_abilene,1,144);
w=gmtm_abilene_vector';
tg=gmtm_abilene;
tg_abilene_vector = reshape(tg, 1, 144);
tg=tg_abilene_vector';
```

```
estimated_traffic_matrix_as_a_vector=wlse_ROUGHAN(A,x,tg,w);
estimated_traffic_matrix=reshape(estimated_traffic_matrix_as_a_vector,12,12);

%calcular error entre matriz original (tm) y estimada
error=tm-estimated_traffic_matrix;
mse=sum(sum(error.^2));
original=sum(sum(tm^2));
estimated_traffic_matrix_error=mse/original;

%guardar el error de cada una de las matrices estimadas
estimated_traffic_matrix_matrices_error=[estimated_traffic_matrix_matrices_error
estimated_traffic_matrix_error];

%guardar todas las matrices estimadas
estimated_traffic_matrix_matrices=[estimated_traffic_matrix_matrices estimated_traffic_matrix];
        end;
   end;
   fprintf(strcat('Day: ',daystring,'\n'));
end;

>> mean(estimated_traffic_matrix_matrices_error)

ans =

    0.41364
```

El error MSE entre las TMs originales y las estimadas es del 41,36% utilizando como vector de pesos (w) la ponderación gravity_model.

A.II.1.4. Análisis del mes 05 de 2004 con ponderación sqrt(gravity_model)

```
format short g    %aumentar precisión
path='C:\Users\ivan\tfc\matlab\abilene-TM\TM\2004\05\TM-2004-05'
link_loads_matrix=[];
tm_vector_matrix=[];
new_routing_matrix=[];

load abileneWeightsInfAutolinks
load abileneAdjacencyAutoLinks
[routing_matrix, sparseroutingmatrix, pathlengths, pathcorrelationmatrix,
pathsparsecorrelationmatrix]=routingMatrix2(abileneWeightsInfAutolinks,abileneAdjacencyAutoli
nks);

%Buscar los enlaces
link_origins=[];
link_destinations=[];
index=1:size(abileneAdjacencyAutolinks,1);
for i=index;
        for j=index;
                if(abileneAdjacencyAutolinks (i,j))
                        link_origins=[link_origins i];
                        link_destinations=[link_destinations j];
                end;
        end;
end;

%Reducir la matriz a los nodos que están presentes
new_routing_matrix=[];
```

```
index=1:length(link_origins);
for i=index;
        row_index=(link_origins(i)-1)*12+ link_destinations(i);% row index
        row_contents=routing_matrix(row_index,:);
        new_routing_matrix=[new_routing_matrix; row_contents];
end;

tm_matrix_matrices=[];
estimated_traffic_matrix_matrices=[];
gmtm_abilene_matrix=[];
estimated_traffic_matrix_matrices_error=[];

for day=1:31;
   daystring=num2str(day,'%02d');
   for hour=0:23;
      hourstring=num2str(hour,'%02d');
      for minutes=0:5:55;
         minutestring=num2str(minutes,'%02d');
         tmpath=strcat(path,'-',daystring,'-',hourstring,minutestring,'.xml');
         tm=AbileneTMparser(tmpath);
          %Reshape:permite convertir una matriz en otra (por ejemplo, una 12x12 en 144x1)
         tm_vector = reshape(tm', 1, 144);
         link_loads = new_routing_matrix *tm_vector';

          %aplicar modelo de gravedad
          gmtm_abilene=GravityModel(tm);
          gmtm_abilene_matrix=[gmtm_abilene_matrix gmtm_abilene];

          %link_loads_matrix=[link_loads_matrix link_loads]; %es útil guardar link_loads_matrix?
          %tm_vector_matrix=[tm_vector_matrix; tm_vector]; %es útil guardar tm_vector_matrix?
          tm_matrix_matrices=[tm_matrix_matrices tm];

%wlse
A=new_routing_matrix;
x=link_loads;

w_as_a_matrix=sqrt(gmtm_abilene);
w_as_a_vector=reshape(w_as_a_matrix,1,144);
w=w_as_a_vector';

tg=gmtm_abilene;
tg_abilene_vector = reshape(tg, 1, 144);
tg=tg_abilene_vector';

estimated_traffic_matrix_as_a_vector=wlse_ROUGHAN(A,x,tg,w);
estimated_traffic_matrix=reshape(estimated_traffic_matrix_as_a_vector,12,12);

%calcular error entre matriz original (tm) y estimada
error=tm-estimated_traffic_matrix;
mse=sum(sum(error.^2));
original=sum(sum(tm^2));
estimated_traffic_matrix_error=mse/original;

%guardar el error de cada una de las matrices estimadas
estimated_traffic_matrix_matrices_error=[estimated_traffic_matrix_matrices_error
estimated_traffic_matrix_error];

%guardar todas las matrices estimadas
estimated_traffic_matrix_matrices=[estimated_traffic_matrix_matrices estimated_traffic_matrix];
        end;
```

```
   end;
   fprintf(strcat('Day: ',daystring,'\n'));
end;


>> mean(estimated_traffic_matrix_matrices_error)

ans =

    0.31947
```

El error MSE entre las TMs originales y las estimadas es del 31,94% utilizando como vector de pesos (w) la ponderación sqrt(gravity_model).

A.II.1.5. Análisis del día 02 del mes 05 de 2004 con ponderación sqrt(gravity_model)

```
path='C:\Users\ivan\tfc\matlab\abilene-TM\TM\2004\05\TM-2004-05'
link_loads_matrix=[];
tm_vector_matrix=[];
new_routing_matrix=[];

load abileneWeightsInfAutolinks
load abileneAdjacencyAutoLinks
[routing_matrix, sparseroutingmatrix, pathlengths, pathcorrelationmatrix,
pathsparsecorrelationmatrix]=routingMatrix2(abileneWeightsInfAutolinks,abileneAdjacencyAutoli
nks);

%Buscar los enlaces
link_origins=[];
link_destinations=[];
index=1:size(abileneAdjacencyAutolinks,1);
for i=index;
        for j=index;
                if(abileneAdjacencyAutolinks (i,j))
                        link_origins=[link_origins i];
                        link_destinations=[link_destinations j];
                end;
        end;
end;


%Reducir la matriz a los nodos que están presentes
new_routing_matrix=[];
index=1:length(link_origins);
for i=index;
        row_index=(link_origins(i)-1)*12+ link_destinations(i);% row index
        row_contents=routing_matrix(row_index,:);
        new_routing_matrix=[new_routing_matrix; row_contents];
end;

tm_matrix_matrices=[];
estimated_traffic_matrix_matrices=[];
gmtm_abilene_matrix=[];
estimated_traffic_matrix_matrices_error=[];

for day=2:2;
   daystring=num2str(day,'%02d');
```

```
   for hour=0:23;
      hourstring=num2str(hour,'%02d');
      for minutes=0:5:55;
         minutestring=num2str(minutes,'%02d');
         tmpath=strcat(path,'-',daystring,'-',hourstring,minutestring,'.xml');
         tm=AbileneTMparser(tmpath);
          %Reshape:permite convertir una matriz en otra (por ejemplo, una 12x12 en 144x1)
         tm_vector = reshape(tm', 1, 144);
         link_loads = new_routing_matrix *tm_vector';
          %aplicar modelo de gravedad
gmtm_abilene=GravityModel(tm);
gmtm_abilene_matrix=[gmtm_abilene_matrix gmtm_abilene];
         %link_loads_matrix=[link_loads_matrix link_loads];
         %tm_vector_matrix=[tm_vector_matrix; tm_vector]; %es útil guardar tm_vector_matrix?
tm_matrix_matrices=[tm_matrix_matrices tm];

%wlse
A=new_routing_matrix;
x=link_loads;
w_as_a_matrix=sqrt(gmtm_abilene);
w_as_a_vector=reshape(w_as_a_matrix,1,144);
w=w_as_a_vector';
tg=gmtm_abilene;

%No se puede multplicar A*tg porque el nº de columnas de A no coincide con el nº de filas de
tg. Hay que convertir tg.
tg_vector = reshape(tg, 1, 144);
tg=tg_vector';

estimated_traffic_matrix_as_a_vector=wlse_ROUGHAN(A,x,tg,w);
estimated_traffic_matrix=reshape(estimated_traffic_matrix_as_a_vector,12,12);

%calcular error entre matriz original (tm) y estimada
error=tm-estimated_traffic_matrix;
mse=sum(sum(error.^2));
original=sum(sum(tm^2));
estimated_traffic_matrix_error=mse/original;

%guardar el error de cada una de las matrices estimadas
estimated_traffic_matrix_matrices_error=[estimated_traffic_matrix_matrices_error
estimated_traffic_matrix_error];

%guardar todas las matrices estimadas
estimated_traffic_matrix_matrices=[estimated_traffic_matrix_matrices estimated_traffic_matrix];
end;
   end;
   fprintf(strcat('Day: ',daystring,'\n'));
end;

>> mean(estimated_traffic_matrix_matrices_error)

ans =

   0.0941
```

En este caso, analizando únicamente el día 02 del mes 05, el error MSE entre las TMs originales y las estimadas es del 9,41% utilizando como vector de pesos (w) la ponderación sqrt(gravity_model).

A.II.2. GÉANT

A.II.2.1. Análisis del día 02 del mes 01 de 2005 con ponderación sqrt(gravity_model)

```
path='C:\Users\ivan\tfc\matlab\traffic-matrices-anonymized-v2\traffic-matrices\IntraTM-2005-01'
link_loads_matrix=[];
tm_vector_matrix=[];
new_routing_matrix=[];

load geantWeightsInfAutolinks
load geantAdjacencyAutoLinks
[routing_matrix, sparseroutingmatrix, pathlengths, pathcorrelationmatrix,
pathsparsecorrelationmatrix]=routingMatrix(geantWeightsInfAutolinks,geantAdjacencyAutoLinks
);

%Buscar los enlaces
link_origins=[];
link_destinations=[];
index=1:size(geantAdjacencyAutoLinks,1);
for i=index;
        for j=index;
                if(geantAdjacencyAutoLinks (i,j))
                        link_origins=[link_origins i];
                        link_destinations=[link_destinations j];
                end;
        end;
end;

%Reducir la matriz a los nodos que están presentes
new_routing_matrix=[];
index=1:length(link_origins);
for i=index;
        row_index=(link_origins(i)-1)*23+ link_destinations(i);% row index
        row_contents=routing_matrix(row_index,:);
        new_routing_matrix=[new_routing_matrix; row_contents];
end;

tm_matrix_matrices=[];
estimated_traffic_matrix_matrices=[];
gmtm_geant_matrix=[];
estimated_traffic_matrix_matrices_error=[];

for day=02:02;
   daystring=num2str(day,'%02d');
   for hour=0:23;
      hourstring=num2str(hour,'%02d');
      for minutes=0:15:45;
         minutestring=num2str(minutes,'%02d');
         tmpath=strcat(path,'-',daystring,'-',hourstring,'-',minutestring,'.xml');
         tm=GeantTMparser(tmpath);
          %Reshape:permite convertir una matriz en otra (por ejemplo, una 12x12 en 144x1)
         tm_vector = reshape(tm', 1, 529);
         link_loads = new_routing_matrix*tm_vector';
          %aplicar modelo de gravedad
gmtm_geant=GravityModel(tm);
gmtm_geant_matrix=[gmtm_geant_matrix gmtm_geant];
        %link_loads_matrix=[link_loads_matrix link_loads];
```

```
        %tm_vector_matrix=[tm_vector_matrix; tm_vector]; %es útil guardar tm_vector_matrix?
tm_matrix_matrices=[tm_matrix_matrices tm];

%wlse
A=new_routing_matrix;
x=link_loads;
w_as_a_matrix=sqrt(gmtm_geant);
w_as_a_vector=reshape(w_as_a_matrix,1,529);
w=w_as_a_vector';
tg=gmtm_geant;
%No se puede multplicar A*tg porque el nº de columnas de A no coincide con el nº de filas de
tg. Hay que convertir tg.
tg_vector = reshape(tg, 1, 529);
tg=tg_vector';

estimated_traffic_matrix_as_a_vector=wlse_ROUGHAN(A,x,tg,w);
estimated_traffic_matrix=reshape(estimated_traffic_matrix_as_a_vector,23,23);

%calcular error entre matriz original (tm) y estimada
error=tm-estimated_traffic_matrix;
mse=sum(sum(error.^2));
original=sum(sum(tm^2));
estimated_traffic_matrix_error=mse/original;

%guardar el error de cada una de las matrices estimadas
estimated_traffic_matrix_matrices_error=[estimated_traffic_matrix_matrices_error
estimated_traffic_matrix_error];

%guardar todas las matrices estimadas
estimated_traffic_matrix_matrices=[estimated_traffic_matrix_matrices estimated_traffic_matrix];
end;
   end;
   fprintf(strcat('Day: ',daystring,'\n'));
end;

>> mean(estimated_traffic_matrix_matrices_error)

ans =

   0.3509
```

Para el día 02 del mes 01 de 2005, el error MSE entre las TMs originales y las estimadas es del 35,09% utilizando como vector de pesos (w) la ponderación sqrt(gravity_model).

A.II.2.2. Análisis del mes 01 de 2005 con ponderación sqrt(gravity_model)

```
path='C:\Users\ivan\tfc\matlab\traffic-matrices-anonymized-v2\traffic-matrices\IntraTM-2005-01'
link_loads_matrix=[];
tm_vector_matrix=[];
new_routing_matrix=[];

load geantWeightsInfAutolinks
load geantAdjacencyAutoLinks
[routing_matrix, sparseroutingmatrix, pathlengths, pathcorrelationmatrix,
pathsparsecorrelationmatrix]=routingMatrix(geantWeightsInfAutolinks,geantAdjacencyAutoLinks
);
```

```
%Buscar los enlaces
link_origins=[];
link_destinations=[];
index=1:size(geantAdjacencyAutoLinks,1);
for i=index;
        for j=index;
                if(geantAdjacencyAutoLinks (i,j))
                        link_origins=[link_origins i];
                        link_destinations=[link_destinations j];
                end;
        end;
end;

%Reducir la matriz a los nodos que están presentes
new_routing_matrix=[];
index=1:length(link_origins);
for i=index;
        row_index=(link_origins(i)-1)*23+ link_destinations(i);% row index
        row_contents=routing_matrix(row_index,:);
        new_routing_matrix=[new_routing_matrix; row_contents];
end;

tm_matrix_matrices=[];
estimated_traffic_matrix_matrices=[];
gmtm_geant_matrix=[];
estimated_traffic_matrix_matrices_error=[];

for day=01:14;
   daystring=num2str(day,'%02d');
   for hour=0:23;
      hourstring=num2str(hour,'%02d');
      for minutes=0:15:45;
         minutestring=num2str(minutes,'%02d');
         tmpath=strcat(path,'-',daystring,'-',hourstring,'-',minutestring,'.xml');
         tm=GeantTMparser(tmpath);
          %Reshape:permite convertir una matriz en otra (por ejemplo, una 12x12 en 144x1)
         tm_vector = reshape(tm', 1, 529);
         link_loads = new_routing_matrix*tm_vector';
          %aplicar modelo de gravedad
gmtm_geant=GravityModel(tm);
gmtm_geant_matrix=[gmtm_geant_matrix gmtm_geant];
        %link_loads_matrix=[link_loads_matrix link_loads];
        %tm_vector_matrix=[tm_vector_matrix; tm_vector]; %es útil guardar tm_vector_matrix?
tm_matrix_matrices=[tm_matrix_matrices tm];

%wlse
A=new_routing_matrix;
x=link_loads;
w_as_a_matrix=sqrt(gmtm_geant);
w_as_a_vector=reshape(w_as_a_matrix,1,529);
w=w_as_a_vector';
tg=gmtm_geant;
%No se puede multplicar A*tg porque el nº de columnas de A no coincide con el nº de filas de
tg. Hay que convertir tg.
tg_vector = reshape(tg, 1, 529);
tg=tg_vector';

estimated_traffic_matrix_as_a_vector=wlse_ROUGHAN(A,x,tg,w);
estimated_traffic_matrix=reshape(estimated_traffic_matrix_as_a_vector,23,23);
```

```
%calcular error entre matriz original (tm) y estimada
error=tm-estimated_traffic_matrix;
mse=sum(sum(error.^2));
original=sum(sum(tm^2));
estimated_traffic_matrix_error=mse/original;

%guardar el error de cada una de las matrices estimadas
estimated_traffic_matrix_matrices_error=[estimated_traffic_matrix_matrices_error
estimated_traffic_matrix_error];

%guardar todas las matrices estimadas
estimated_traffic_matrix_matrices=[estimated_traffic_matrix_matrices estimated_traffic_matrix];
end;
   end;
   fprintf(strcat('Day: ',daystring,'\n'));
end;

>> mean(estimated_traffic_matrix_matrices_error)

ans =

   0.3138
```

Para el mes 01 de 2005, el error MSE entre las TMs originales y las estimadas es del 31,38% utilizando como vector de pesos (w) la ponderación sqrt(gravity_model).

A.II.3. Comprobaciones

A lo largo del estudio se realizaron varias comprobaciones para validar los resultados obtenidos mediante *wlse.*

A.II.3.1. Función reshape

Comprobar que se siguen manteniendo los valores iniciales al aplicar la función reshape a “tg” en *wlse.*

Cargar una matriz

```
>>    tm1_abilene=abileneTMparser('C:\Users\ivan\tfc\matlab\abilene-TM\TM\2004\09\TM-2004-
09-06-1645.xml')
```

Aplicar GM

```
>> gmtm1_abilene=GravityModel(tm1_abilene);
>> tg=gmtm1_abilene; %matriz de dimensiones (12, 12)
>> tg1_abilene_vector = reshape(tg, 1, 144);
>> tg=tg1_abilene_vector'; %vector
>> tg_matrix_12x12 = reshape(tg, 12, 12); %de nuevo matriz de dimensiones (12, 12)

>> help isequal

   ISEQUAL(A,B) is 1 if the two arrays are the same size
   and contain the same values, and 0 otherwise.
```

```
   ISEQUAL(A,B,C,...) is 1 if all the input arguments are
   numerically equal.

   ISEQUAL recursively compares the contents of cell arrays and
   structures.  If all the elements of a cell array or structure
   are numerically equal, ISEQUAL will return 1.

   If B is defined, and you set A = B; then ISEQUAL(A,B) is
   not necessarily true. If B contains a NaN element, NaNs
   are not equal to each other by definition, and ISEQUAL
   will return false.

>> isequal (gmtm1_abilene, tg_matrix_12x12)

ans =

    1
```

A.II.3.2. Consistencia de los resultados aplicando ponderación constante

En el punto A.II.1.1 del anexo se ha calculado la matriz estimada de TM-2004-09-06-1645.xml con w=ones(144, 1). El error ha sido del 4,38%.
Para verificar la consistencia de los resultados con la ponderación constante, se aplica *wlse* con w=15 * ones (144, 1).

```
>> A=new_routing_matrix;
>> x=link_loads;
>> w=15 * ones(144,1); %ponderación de pesos constante
>> tg=gmtm1_abilene;
>> tg1_abilene_vector = reshape(tg, 1, 144);
>> tg=tg1_abilene_vector'
>> estimated_traffic_matrix_as_a_vector=wlse_ROUGHAN(A,x,tg,w)
>> estimated_traffic_matrix=reshape(estimated_traffic_matrix_as_a_vector,12,12)

>> imagesc(estimated_traffic_matrix);colorbar;
>> error=tm1_abilene-estimated_traffic_matrix;
>> imagesc(error);colorbar
>> mse=sum(sum(error.^2))
original=sum(sum(tm1_abilene^2))
mse/original

mse =

  7.0669e+010


original =

  1.6122e+012


ans =

   0.0438
```

El resultado es idéntico al obtenido con w=ones(144, 1), un 4,38% de error entre la TM original (TM-2004-09-06-1645.xml) y la estimada.

A.III. RRDtool

Los conceptos básicos para entender la arquitectura de una base de datos RRD son los siguientes:

> **"DS** is a key word. variable_name is a name under which the parameter is saved in the database. There can be as many DSs in a database as needed. After every step interval, a new value of DS is supplied to update the database. This value is also called Primary Data Point **(PDP)**." http://oss.oetiker.ch/rrdtool/tut/rrd-beginners.en.html

> "RRDtool assumes time-variable data in intervals of a certain length. This interval, usually named step, is specified upon creation of an RRD file and cannot be changed afterwards. Because data may not always be available at just the right time, RRDtool will automatically interpolate any submitted data to fit its internal time-steps.
> The value for a specific step, that has been interpolated, is named a primary data point (PDP). Multiple primary data points may be consolidated according to a consolidation function (CF) to form a consolidated data point (CDP). Typical consolidation functions are average, minimum, maximum.
> After the data have been consolidated, the resulting CDP is stored in a round-robin archive (RRA). A round-robin archive stores a fixed amount of CDPs and specifies how many PDPs should be consolidated into one CDP and which CF to use. The total time covered by an RRA can be calculated as follows:
>
> time covered = (#CDPs stored) * (#PDPs per CDP) * step"
>
> http://en.wikipedia.org/wiki/RRDtool

A.III.1. Cabecera de un fichero RRD

Para acceder a la información de la cabecera de un fichero RRD se utiliza el comando *rrdtool info*:

```
ivan@linux-2zba:~> rrdtool info cat-nac.rrd
filename = "cat-nac.rrd"
rrd_version = "0003"
step = 300
last_update = 1271157208
ds[ds0].type = "COUNTER"
ds[ds0].minimal_heartbeat = 600
ds[ds0].min = 0,0000000000e+00
ds[ds0].max = 1,2500000000e+09
ds[ds0].last_ds = "3086237113054704"
ds[ds0].value = 3,9307405892e+10
ds[ds0].unknown_sec = 0
ds[ds1].type = "COUNTER"
ds[ds1].minimal_heartbeat = 600
ds[ds1].min = 0,0000000000e+00
ds[ds1].max = 1,2500000000e+09
ds[ds1].last_ds = "2704827196804179"
ds[ds1].value = 7,5849125198e+10
ds[ds1].unknown_sec = 0
rra[0].cf = "AVERAGE"
```

```
rra[0].rows = 599
rra[0].cur_row = 3
rra[0].pdp_per_row = 1
rra[0].xff = 5,0000000000e-01
rra[0].cdp_prep[0].value = 1,8467583200e+08
rra[0].cdp_prep[0].unknown_datapoints = 0
rra[0].cdp_prep[1].value = 3,1890215400e+08
rra[0].cdp_prep[1].unknown_datapoints = 0
rra[1].cf = "AVERAGE"
rra[1].rows = 700
rra[1].cur_row = 172
rra[1].pdp_per_row = 6
rra[1].xff = 5,0000000000e-01
rra[1].cdp_prep[0].value = 3,5872472846e+08
rra[1].cdp_prep[0].unknown_datapoints = 0
rra[1].cdp_prep[1].value = 7,1279969922e+08
rra[1].cdp_prep[1].unknown_datapoints = 0
rra[2].cf = "AVERAGE"
rra[2].rows = 775
rra[2].cur_row = 680
rra[2].pdp_per_row = 24
rra[2].xff = 5,0000000000e-01
rra[2].cdp_prep[0].value = 2,3688486913e+09
rra[2].cdp_prep[0].unknown_datapoints = 0
rra[2].cdp_prep[1].value = 5,2866507557e+09
rra[2].cdp_prep[1].unknown_datapoints = 0
rra[3].cf = "AVERAGE"
rra[3].rows = 796
rra[3].cur_row = 11
rra[3].pdp_per_row = 288
rra[3].xff = 5,0000000000e-01
rra[3].cdp_prep[0].value = 1,3698772474e+10
rra[3].cdp_prep[0].unknown_datapoints = 5
rra[3].cdp_prep[1].value = 2,1004476876e+10
rra[3].cdp_prep[1].unknown_datapoints = 5
rra[4].cf = "MAX"
rra[4].rows = 600
rra[4].cur_row = 396
rra[4].pdp_per_row = 1
rra[4].xff = 5,0000000000e-01
rra[4].cdp_prep[0].value = 1,8645058400e+08
rra[4].cdp_prep[0].unknown_datapoints = 0
rra[4].cdp_prep[1].value = 3,1390772700e+08
rra[4].cdp_prep[1].unknown_datapoints = 0
rra[5].cf = "MAX"
rra[5].rows = 700
rra[5].cur_row = 296
rra[5].pdp_per_row = 6
rra[5].xff = 5,0000000000e-01
rra[5].cdp_prep[0].value = 1,8114028088e+08
rra[5].cdp_prep[0].unknown_datapoints = 0
rra[5].cdp_prep[1].value = 3,6130730261e+08
rra[5].cdp_prep[1].unknown_datapoints = 0
rra[6].cf = "MAX"
rra[6].rows = 775
rra[6].cur_row = 638
rra[6].pdp_per_row = 24
rra[6].xff = 5,0000000000e-01
rra[6].cdp_prep[0].value = 1,9615666487e+08
rra[6].cdp_prep[0].unknown_datapoints = 0
rra[6].cdp_prep[1].value = 4,0478574195e+08
```

```
rra[6].cdp_prep[1].unknown_datapoints = 0
rra[7].cf = "MAX"
rra[7].rows = 796
rra[7].cur_row = 454
rra[7].pdp_per_row = 288
rra[7].xff = 5,0000000000e-01
rra[7].cdp_prep[0].value = 2,0535758833e+08
rra[7].cdp_prep[0].unknown_datapoints = 5
rra[7].cdp_prep[1].value = 4,0478574195e+08
rra[7].cdp_prep[1].unknown_datapoints = 5
```

A.III.2. Conversión de un fichero RRD a XML

Para convertir un fichero RRD al formato XML se utiliza el comando *rrdtool dump*:

```
ivan@linux-2zba:~> rrdtool dump cat-nac.rrd > cat-nac.xml
```

```
<rra>
  <cf>AVERAGE</cf>
  <pdp_per_row>1</pdp_per_row>
  - <!--
   300 seconds
    -->
<params>
  <xff>5.0000000000e-01</xff>
    </params>
<cdp_prep>
<ds>
  <primary_value>1.8114028088e+08</primary_value>
  <secondary_value>NaN</secondary_value>
  <value>1.8467583200e+08</value>
  <unknown_datapoints>0</unknown_datapoints>
    </ds>
<ds>
  <primary_value>3.5935852374e+08</primary_value>
  <secondary_value>NaN</secondary_value>
  <value>3.1890215400e+08</value>
  <unknown_datapoints>0</unknown_datapoints>
    </ds>
    </cdp_prep>
<database>
  - <!--
   2010-04-11 11:20:00 CEST / 1270977600
    -->
<row>
  <v>8.0773437249e+07</v>
  <v>6.8187138659e+07</v>
    </row>
    .
    .
    .
  </row>
  - <!--
```

```
    2010-04-13 13:10:00 CEST / 1271157000
     -->
<row>
  <v>1.8114028088e+08</v>
  <v>3.5935852374e+08</v>
    </row>
    </database>
    </rra>
<rra>
  <cf>AVERAGE</cf>
  <pdp_per_row>6</pdp_per_row>
  - <!--
    1800 seconds
     -->
<params>
  <xff>5.0000000000e-01</xff>
    </params>
<cdp_prep>
<ds>
  <primary_value>1.6761775143e+08</primary_value>
  <secondary_value>1.7998928153e+08</secondary_value>
  <value>3.5872472846e+08</value>
  <unknown_datapoints>0</unknown_datapoints>
    </ds>
<ds>
  <primary_value>3.8692398124e+08</primary_value>
  <secondary_value>3.6130730261e+08</secondary_value>
  <value>7.1279969922e+08</value>
  <unknown_datapoints>0</unknown_datapoints>
    </ds>
    </cdp_prep>
<database>
  - <!--
    2010-03-29 23:30:00 CEST / 1269898200
     -->
<row>
  <v>9.6764190890e+07</v>
  <v>8.8485652830e+07</v>
    </row>
     .
     .
     .
  - <!--
    2010-04-13 13:00:00 CEST / 1271156400
     -->
<row>
  <v>1.6761775143e+08</v>
  <v>3.8692398124e+08</v>
    </row>
    </database>
    </rra>
<rra>
  <cf>AVERAGE</cf>
  <pdp_per_row>24</pdp_per_row>
```

```
  - <!--
    7200 seconds
    -->
<params>
  <xff>5.0000000000e-01</xff>
    </params>
<cdp_prep>
<ds>
  <primary_value>1.4307373175e+08</primary_value>
  <secondary_value>1.9615666487e+08</secondary_value>
  <value>2.3688486913e+09</value>
  <unknown_datapoints>0</unknown_datapoints>
    </ds>
<ds>
  <primary_value>3.4373146751e+08</primary_value>
  <secondary_value>3.6033560412e+08</secondary_value>
  <value>5.2866507557e+09</value>
  <unknown_datapoints>0</unknown_datapoints>
    </ds>
    </cdp_prep>
<database>
  - <!--
    2010-02-07 23:00:00 CET / 1265580000
    -->
<row>
  <v>1.7553559189e+08</v>
  <v>9.7833356766e+07</v>
    </row>
    .
    .
    .
  - <!--
    2010-04-13 12:00:00 CEST / 1271152800
    -->
- <row>
  <v>1.4307373175e+08</v>
  <v>3.4373146751e+08</v>
    </row>
    </database>
    </rra>
<rra>
  <cf>AVERAGE</cf>
  <pdp_per_row>288</pdp_per_row>
  - <!--
    86400 seconds
    -->
<params>
  <xff>5.0000000000e-01</xff>
    </params>
<cdp_prep>
<ds>
  <primary_value>1.4053461328e+08</primary_value>
  <secondary_value>1.1024178841e+08</secondary_value>
  <value>1.3698772474e+10</value>
```

```
  <unknown_datapoints>5</unknown_datapoints>
    </ds>
<ds>
  <primary_value>1.7906883881e+08</primary_value>
  <secondary_value>6.3501744665e+07</secondary_value>
  <value>2.1004476876e+10</value>
  <unknown_datapoints>5</unknown_datapoints>
    </ds>
    </cdp_prep>
<database>
  - <!--
    2008-02-08 01:00:00 CET / 1202428800
    -->
<row>
  <v>1.9525561300e+08</v>
  <v>8.1915781000e+07</v>
    </row>
     .
     .
     .
   </row>
    <!--
    2010-04-13 02:00:00 CEST / 1271116800
    -->
 <row>
  <v>1.4053461328e+08</v>
  <v>1.7906883881e+08</v>
    </row>
    </database>
    </rra>
```

La base de datos cat-nac.rrd está formada por cuatro RRAs, Round Robin Archive. Al principio de cada RRA está la etiqueta <pdp_per_row> con el número de PDPs que se consolidarán en cada CDP. Para el caso de la RRA de 1.800 segundos, este valor es 6.

```
    <rra>
      <cf>AVERAGE</cf>
      <pdp_per_row>6</pdp_per_row>
      - <!--
        1800 seconds
      -->
```

Del fichero XML también se extrae la información sobre las fechas de inicio y finalización de los CDPs:

- En la RRA de 300 segundos están guardados los valores comprendidos entre
  2010-04-11 11:20:00 CEST / 1270977600
  2010-04-13 13:10:00 CEST / 1271157000

- En la RRA de 1.800 segundos están guardados los valores comprendidos entre
  2010-03-29 23:30:00 CEST / 1269898200
  2010-04-13 13:00:00 CEST / 1271156400

- En la RRA de 7.200 segundos están guardados los valores comprendidos entre
  2010-02-07 23:00:00 CET / 1265580000
  2010-04-13 12:00:00 CEST / 1271152800

- En la RRA de 86.400 segundos (1 día) están guardados los valores comprendidos entre
  2008-02-08 01:00:00 CET / 1202428800
  2010-04-13 02:00:00 CEST / 1271116800

A.III.3. Herramientas para obtener información de un fichero RRD

Aparte de *info* y *dump*, existen otras herramientas adicionales incluidas en RRDtool para obtener información de un fichero RRD.

A.III.3.1. fetch

Con el siguiente comando se extraen de la RRA todos los valores almacenados durante el mes de enero de 2010.

```
ivan@linux-2zba:~> rrdtool fetch cat-nac.rrd AVERAGE -s 20100101 -e
start+31d
                 ds0                 ds1

1262304000: 8,9237905265e+07 5,9683532502e+07
1262390400: 8,7332470151e+07 8,7767620292e+07
1262476800: 9,3183359024e+07 5,6552064598e+07
1262563200: 9,6178987896e+07 6,9257060603e+07
1262649600: 1,0982343003e+08 9,2851339372e+07
1262736000: 1,2461345232e+08 9,7978279870e+07
1262822400: 1,0831350631e+08 1,3043795617e+08
1262908800: 1,2153551367e+08 1,5101775941e+08
1262995200: 1,3164030330e+08 1,9076372330e+08
1263081600: 1,3182394406e+08 1,0570924874e+08
1263168000: 1,1997561922e+08 7,2042679623e+07
1263254400: 1,6747569589e+08 1,5833009441e+08
1263340800: 1,5861058936e+08 1,5706972000e+08
1263427200: 1,8509666284e+08 1,5272968671e+08
1263513600: 1,8811626105e+08 1,6441266963e+08
1263600000: 1,9734257340e+08 1,5572697156e+08
1263686400: 1,5918503739e+08 7,8172921865e+07
1263772800: 1,2078915356e+08 6,7797295081e+07
1263859200: 1,6132468068e+08 1,5939926638e+08
1263945600: 1,7110816879e+08 1,6179906079e+08
1264032000: 1,6214381918e+08 1,5594068522e+08
1264118400: 1,6180900565e+08 1,6276515206e+08
1264204800: 1,6675334204e+08 1,4712874244e+08
1264291200: 1,1694253817e+08 8,0379432775e+07
```

```
1264377600: 1,2120649946e+08 7,1286957201e+07
1264464000: 1,7408641684e+08 1,5918062201e+08
1264550400: 1,7461662215e+08 1,6992335238e+08
1264636800: 1,6008600923e+08 1,4706569217e+08
1264723200: 1,4980276607e+08 1,3789545506e+08
1264809600: 2,0060327409e+08 1,3959983408e+08
1264896000: 1,3679592761e+08 7,2263869235e+07
1264982400: 1,0080762312e+08 6,5506826273e+07
```

Un CDP por día, el valor medio de un día porque la CF es AVERAGE.
La fecha en Unix timestamp correspondiente al primer y último valor:

1262304000 → **GMT**: Fri, 01 Jan 2010 00:00:00 GMT
1264982400 → **GMT**: Mon, 01 Feb 2010 00:00:00 GMT

```
ivan@linux-2zba:~> rrdtool fetch cat-nac.rrd AVERAGE -s 20100101 -e
start+1h
                 ds0                 ds1

1262304000: 8,9237905265e+07 5,9683532502e+07
1262390400: 8,7332470151e+07 8,7767620292e+07
```

A.III.3.2. xport

Con *xport* se obtiene una representación formateada en XML de la base de datos. En el siguiente comando se le pasa a *xport* los DS *ds0* y *ds1* para que represente la primera hora del 1 de enero de 2010; inicio `1262304000` (00:00 horas del 1 de enero de 2010) final `1262307600` (01:00 horas del 1 de enero de 2010)

```
ivan@linux-2zba:~> rrdtool xport -s 1262304000 -e 1262307600 --step
300 DEF:in=cat-nac.rrd:ds0:AVERAGE DEF:out=cat-nac.rrd:ds1:AVERAGE
CDEF:aa=in,out,+,8,* XPORT:in:"in bytes" XPORT:aa:"in and out bits"
<?xml version="1.0" encoding="ISO-8859-1"?>

<xport>
  <meta>
    <start>1262390400</start>
    <step>86400</step>
    <end>1262304000</end>
    <rows>0</rows>
    <columns>2</columns>
    <legend>
      <entry>in bytes</entry>
      <entry>in and out bits</entry>
    </legend>
  </meta>
  <data>
  </data>
</xport>
```

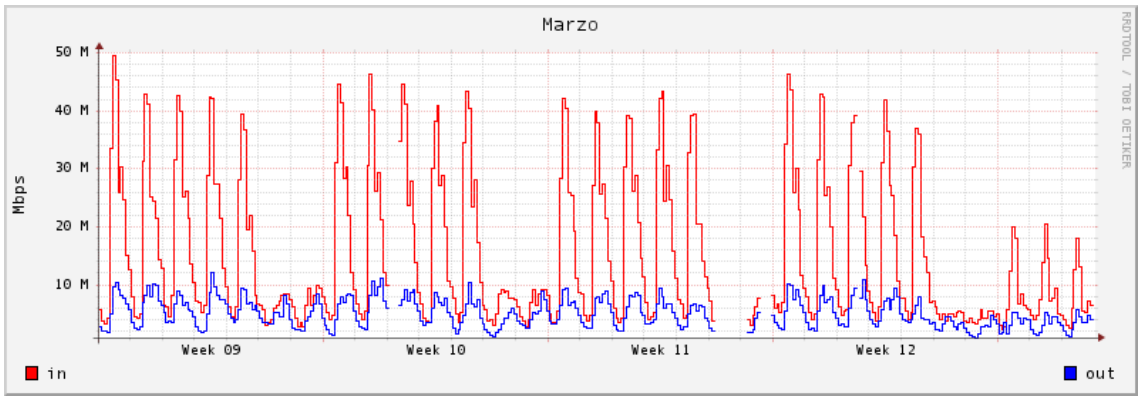
El step es de 86.400 segundos, 1 día.

A.III.4. Generación de gráficos

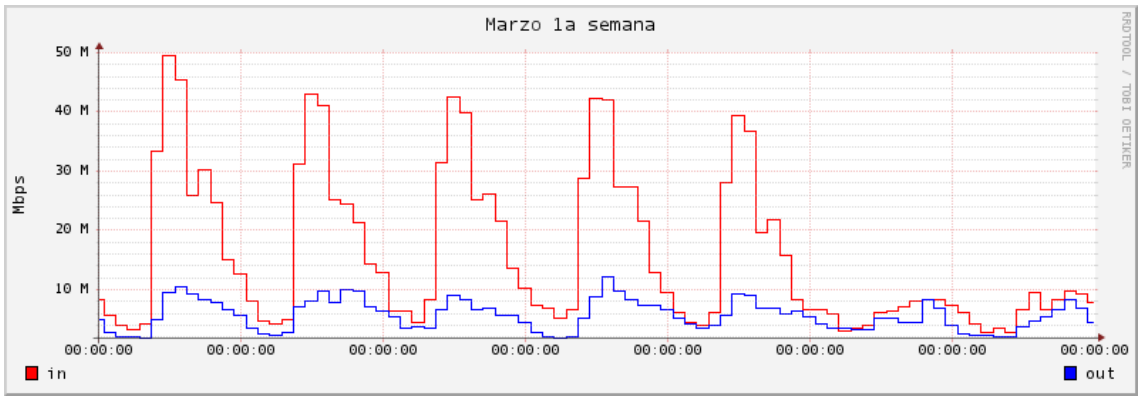
Los siguientes gráficos son una recopilación de aquellos más representativos, ya sea por el formato definido o por los datos que contienen; en todos ellos está escrito el comando utilizado para su creación.

A.III.4.1. and-can-1.rdd

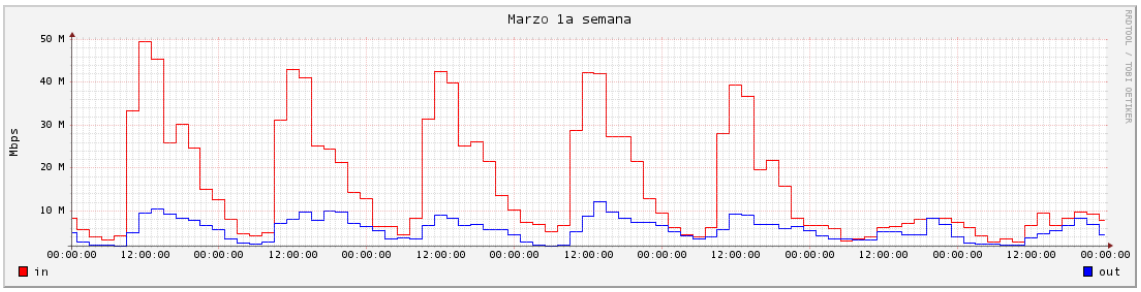
```
rrdtool graph graph_marzo_700x200.png --start 20100301 --end start+31d
--title "Marzo" --height=200 --width=700 --vertical-label="Mbps"
DEF:in=and-can-1.rrd:ds0:AVERAGE LINE:in#ff0000:in DEF:out=and-can-
1.rrd:ds1:AVERAGE LINE:out#0000ff:out
```



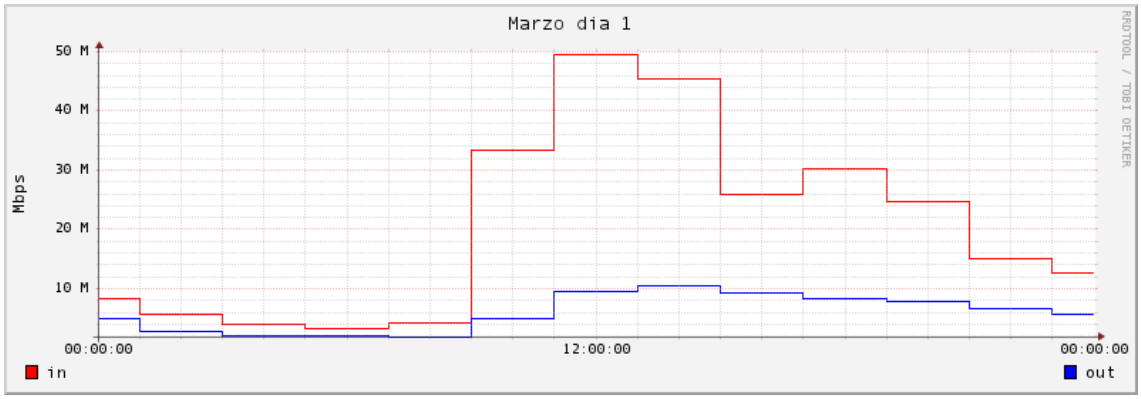

```
rrdtool graph graph_marzo_7d.png --start 20100301 --end start+7d -x
MINUTE:7200:HOUR:24:HOUR:24:0:%X --title "Marzo 1a semana" --
height=200 --width=700 --vertical-label="Mbps" DEF:in=and-can-
1.rrd:ds0:AVERAGE LINE:in#ff0000:in DEF:out=and-can-1.rrd:ds1:AVERAGE
LINE:out#0000ff:out
```



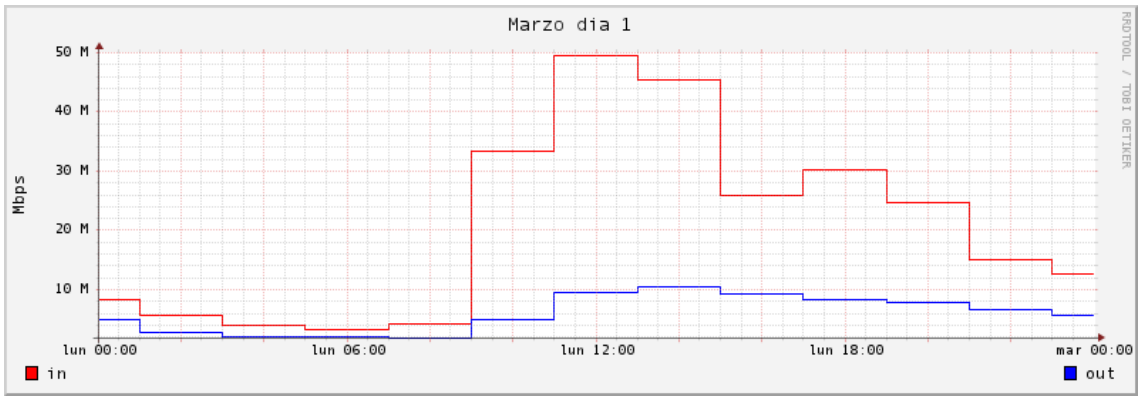

```
rrdtool graph graph_marzo_7d_1000x200.png --start 20100301 --end
start+7d -x MINUTE:60:HOUR:24:HOUR:12:0:%X --title "Marzo 1a semana" -
-height=200 --width=1000 --vertical-label="Mbps" DEF:in=and-can-
1.rrd:ds0:AVERAGE LINE:in#ff0000:in DEF:out=and-can-1.rrd:ds1:AVERAGE
LINE:out#0000ff:out
```




```
rrdtool graph graph_marzo_1d.png --start 20100301 --end start+1d -x
MINUTE:60:HOUR:24:HOUR:12:0:%X --title "Marzo dia 1" --height=200 --
width=700 --vertical-label="Mbps" DEF:in=and-can-l.rrd:ds0:AVERAGE
LINE:in#ff0000:in DEF:out=and-can-l.rrd:ds1:AVERAGE
LINE:out#0000ff:out
```



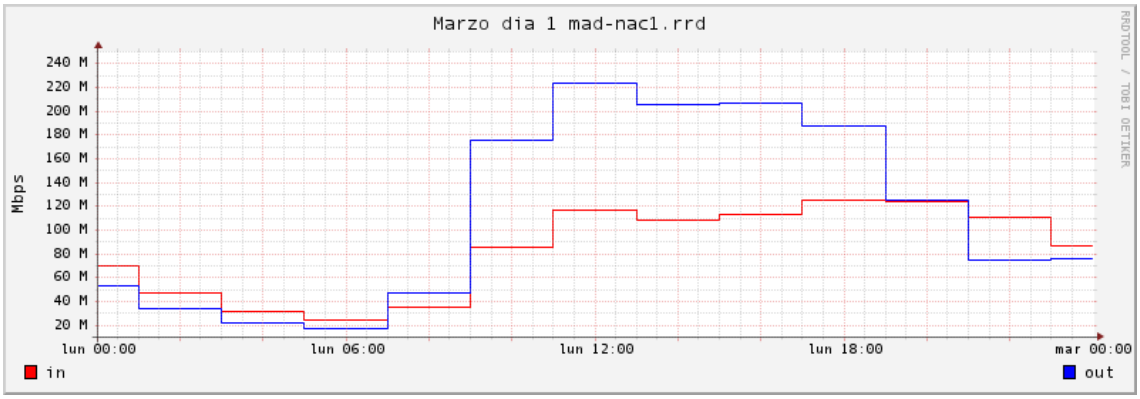

```
rrdtool graph graph_marzo_1d_no-x-grid.png --start 20100301 --end
start+1d --title "Marzo dia 1" --height=200 --width=700 --vertical-
label="Mbps" DEF:in=and-can-l.rrd:ds0:AVERAGE LINE:in#ff0000:in
DEF:out=and-can-l.rrd:ds1:AVERAGE LINE:out#0000ff:out
```



A.III.4.2. mad-nac1.rrd

```
rrdtool graph graph_mad-nac1.rrd_marzo_1d_no-x-grid.png --start
20100301 --end start+1d --title "Marzo dia 1 mad-nac1.rrd" --
height=200 --width=700 --vertical-label="Mbps" DEF:in=mad-
```

```
nac1.rrd:ds0:AVERAGE LINE:in#ff0000:in DEF:out=mad-
nac1.rrd:ds1:AVERAGE LINE:out#0000ff:out
```



"X-Axis

[-x|--x-grid GTM:GST:MTM:MST:LTM:LST:LPR:LFM]
[-x|--x-grid none]

The x-axis label is quite complex to configure. If you don't have very special needs it is probably best to rely on the auto configuration to get this right. You can specify the string none to suppress the grid and labels altogether.
The grid is defined by specifying a certain amount of time in the ?TM positions. You can choose from SECOND, MINUTE, HOUR, DAY, WEEK, MONTH or YEAR. Then you define how many of these should pass between each line or label. This pair (?TM:?ST) needs to be specified for the base grid (G??), the major grid (M??) and the labels (L??). For the labels you also must define a precision in LPR and a strftime format string in LFM. LPR defines where each label will be placed. If it is zero, the label will be placed right under the corresponding line (useful for hours, dates etcetera). If you specify a number of seconds here the label is centered on this interval (useful for Monday, January etcetera).

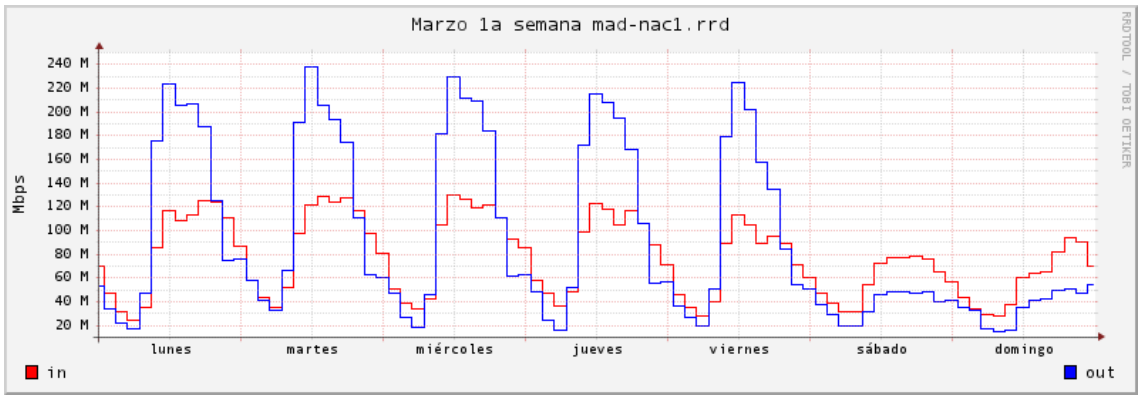
```
 --x-grid MINUTE:10:HOUR:1:HOUR:4:0:%X
```

This places grid lines every 10 minutes, major grid lines every hour, and labels every 4 hours. The labels are placed under the major grid lines as they specify exactly that time.

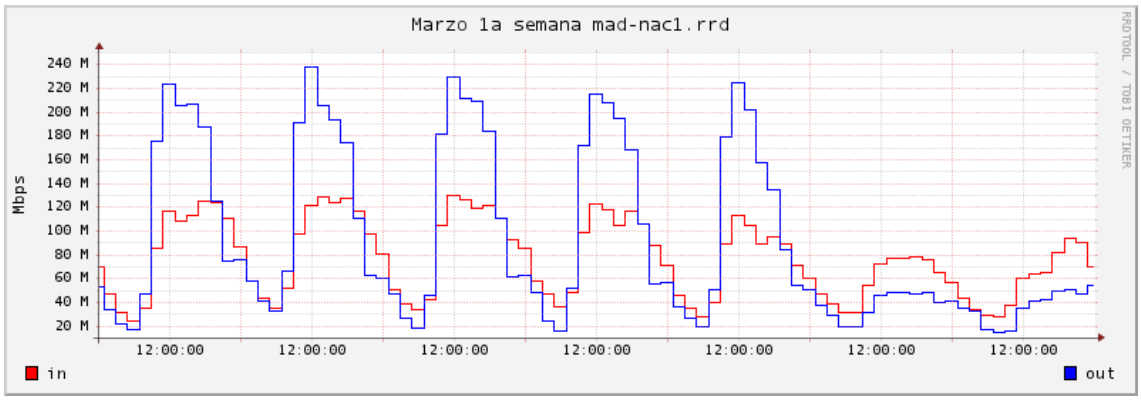
```
 --x-grid HOUR:8:DAY:1:DAY:1:86400:%A
```

This places grid lines every 8 hours, major grid lines and labels each day. The labels are placed exactly between two major grid lines as they specify the complete day and not just midnight."
http://oss.oetiker.ch/rrdtool/doc/rrdgraph.en.html

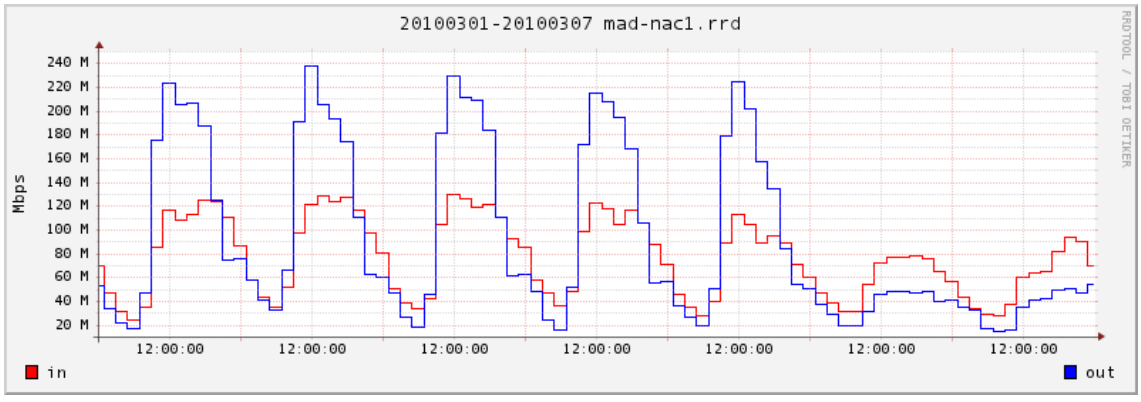
```
rrdtool graph graph_mad-nac1.rrd_marzo_1a_semana.png --start 20100301
--end start+7d -x HOUR:12:DAY:1:DAY:1:86400:%A --title "Marzo 1a
semana mad-nac1.rrd" --height=200 --width=700 --vertical-label="Mbps"
DEF:in=mad-nac1.rrd:ds0:AVERAGE LINE:in#ff0000:in DEF:out=mad-
nac1.rrd:ds1:AVERAGE LINE:out#0000ff:out
```




```
rrdtool graph graph_mad-nac1.rrd_marzo_1a_semana_grid%X.png --start
20100301 --end start+7d -x HOUR:12:DAY:1:DAY:1:86400:%X --title "Marzo
1a semana mad-nac1.rrd" --height=200 --width=700 --vertical-
label="Mbps" DEF:in=mad-nac1.rrd:ds0:AVERAGE LINE:in#ff0000:in
DEF:out=mad-nac1.rrd:ds1:AVERAGE LINE:out#0000ff:out
```




```
rrdtool graph graph_mad-nac1.rrd_marzo_1a_semana_grid%X_2.png --start
20100301 --end start+7d -x HOUR:12:DAY:1:DAY:1:86400:%X --title
"20100301-20100307 mad-nac1.rrd" --height=200 --width=700 --vertical-
label="Mbps" DEF:in=mad-nac1.rrd:ds0:AVERAGE LINE:in#ff0000:in
DEF:out=mad-nac1.rrd:ds1:AVERAGE LINE:out#0000ff:out
```



"Escaping the colon

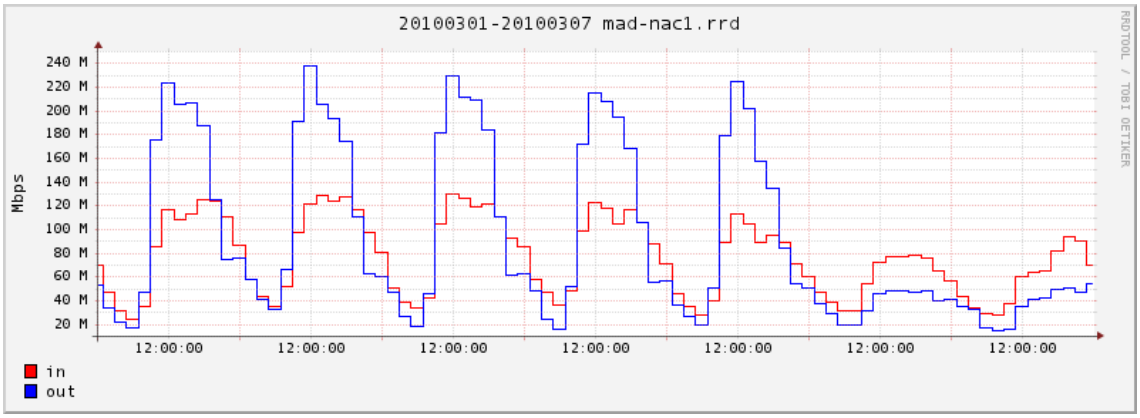
In a ':' in a *legend* argument will mark the end of the legend. To enter a ':' into a legend, the colon must be escaped with a backslash '\:'. Beware, that many environments look for backslashes themselves, so it may be necessary to write two backslashes so that one is passed onto **rrdgraph**.

String Formatting

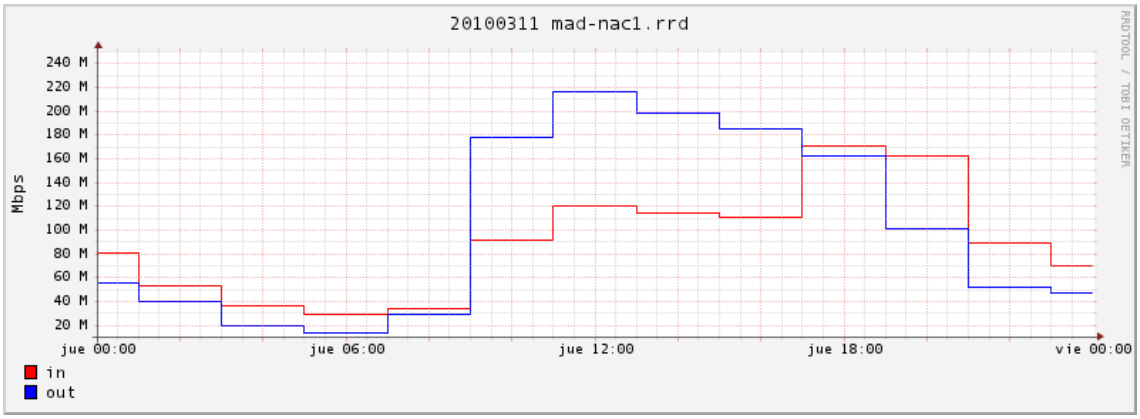
The text printed below the actual graph can be formated by appending special escaped characters at the end of a text. When ever such a character occurs, all pending text is pushed onto the graph according to the character specified.

Valid markers are: **\j** for justified, **\l** for left aligned, **\r** for right aligned and **\c** for centered. In the next section there is an example showing how to use centered formating." http://oss.oetiker.ch/rrdtool/doc/rrdgraph_graph.en.html

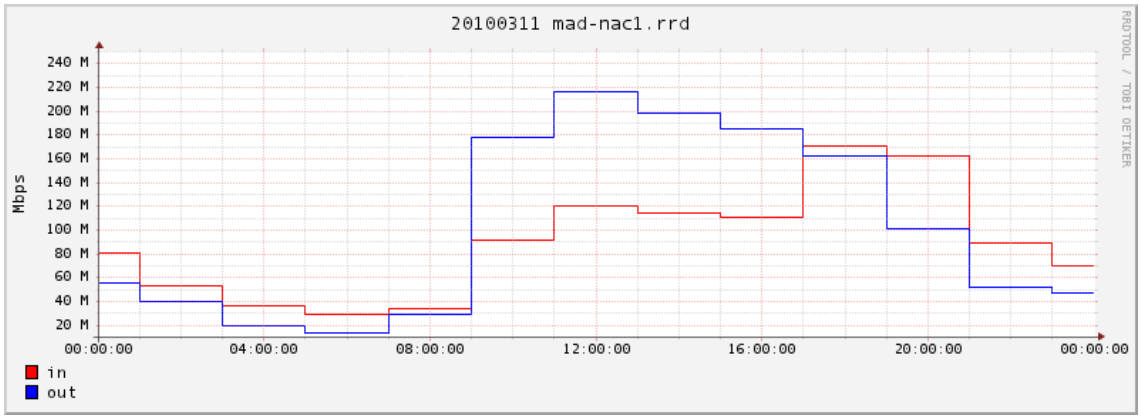
```
rrdtool graph graph_mad-nac1.rrd_marzo_1a_semana_grid%X_2.png --start
20100301 --end start+7d -x HOUR:12:DAY:1:DAY:1:86400:%X --title
"20100301-20100307 mad-nac1.rrd" --height=200 --width=700 --vertical-
label="Mbps" DEF:in=mad-nac1.rrd:ds0:AVERAGE LINE:in#ff0000:in\\n
DEF:out=mad-nac1.rrd:ds1:AVERAGE LINE:out#0000ff:out\\l
```



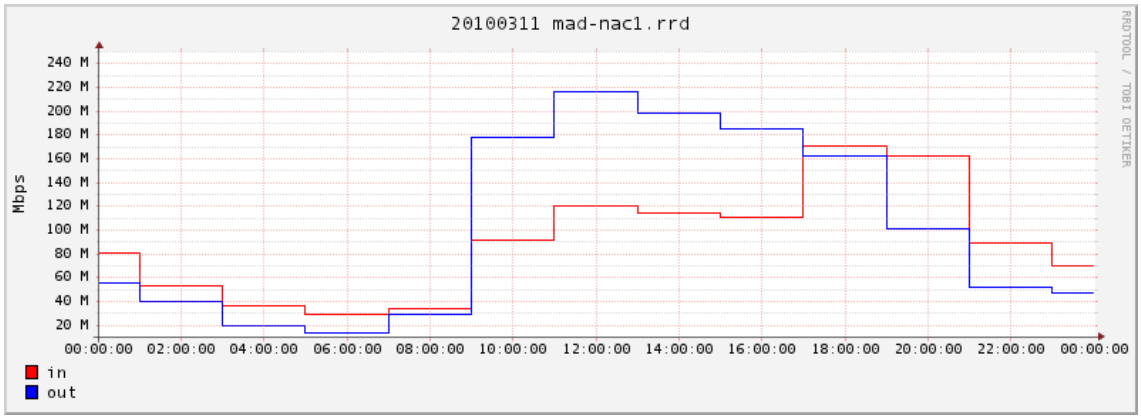

```
rrdtool graph graph_mad-nac1.rrd_20100311.png --start 20100311 --end
start+1d --title "20100311 mad-nac1.rrd" --height=200 --width=700 --
vertical-label="Mbps" DEF:in=mad-nac1.rrd:ds0:AVERAGE
LINE:in#ff0000:in\\n DEF:out=mad-nac1.rrd:ds1:AVERAGE
LINE:out#0000ff:out\\l
```



```
rrdtool graph graph_mad-nac1.rrd_20100311_x_grid.png --start 20100311
--end start+1d -x MINUTE:60:HOUR:4:HOUR:4:0:%X --title "20100311 mad-
nac1.rrd" --height=200 --width=700 --vertical-label="Mbps" DEF:in=mad-
nac1.rrd:ds0:AVERAGE LINE:in#ff0000:in\\n DEF:out=mad-
nac1.rrd:ds1:AVERAGE LINE:out#0000ff:out\\l
```

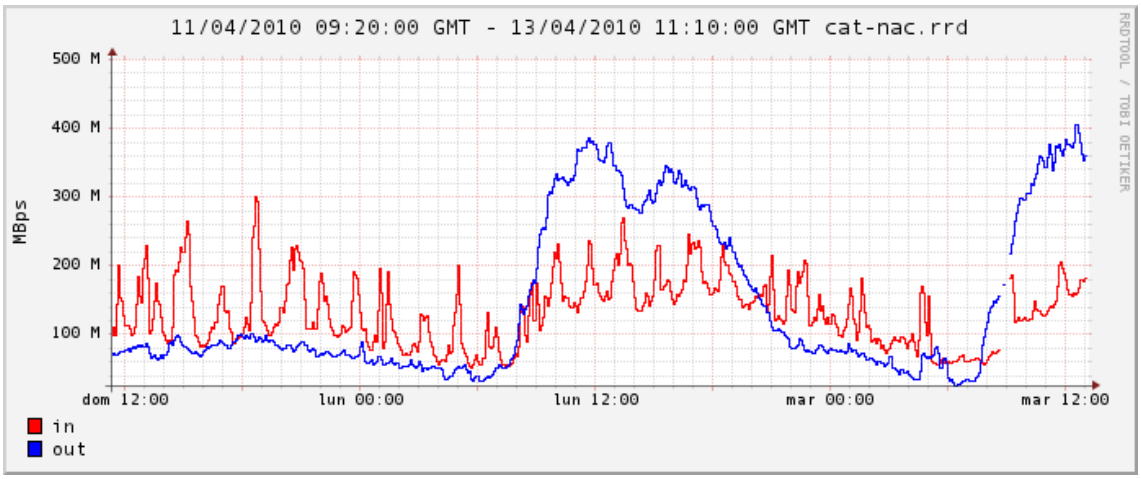



```
rrdtool graph graph_mad-nac1.rrd_20100311_x_grid.png --start 20100311
--end start+1d -x MINUTE:120:HOUR:2:HOUR:2:0:%X --title "20100311 mad-
nac1.rrd" --height=200 --width=700 --vertical-label="Mbps" DEF:in=mad-
nac1.rrd:ds0:AVERAGE LINE:in#ff0000:in\\n DEF:out=mad-
nac1.rrd:ds1:AVERAGE LINE:out#0000ff:out\\l
```



A.III.4.3. cat-nac.rrd

```
rrdtool graph graph_cat-nac.rrd_300s.png --start 1270977600 --end
1271157000 --title "11/04/2010 09:20:00 GMT - 13/04/2010 11:10:00 GMT
cat-nac.rrd" --height=200 --width=1000 --vertical-label="MBps"
DEF:in=cat-nac.rrd:ds0:AVERAGE LINE:in#ff0000:in\\n DEF:out=cat-
nac.rrd:ds1:AVERAGE LINE:out#0000ff:out\\l
```




```
rrdtool graph graph_cat-nac.rrd_20080101_x_grid.png --start 20080101 -
-end now --title "cat-nac.rrd" --height=200 --width=1000 --vertical-
label="MBps" DEF:in=cat-nac.rrd:ds0:AVERAGE LINE:in#ff0000:in\\n
DEF:out=cat-nac.rrd:ds1:AVERAGE LINE:out#0000ff:out\\l
```